The Thesis Committee for Brad Ernest Munoz
Certifies that this is the approved version of the following thesis:

Reliability Methods in Dynamic System Analysis

APPROVED BY

SUPERVISING COMMITTEE:

Raul G. Longoria, Supervisor

Eric P. Fahrenthold

Reliability Methods in Dynamic System Analysis

by

Brad Ernest Munoz, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2012

Reliability Methods in Dynamic System Analysis

Brad Ernest Munoz, M.S.E.
The University of Texas at Austin, 2012

Supervisor: Raul G. Longoria

Standard techniques used to analyze a system's response with uncertain system parameters or inputs, are generally Importance sampling methods. Sampling methods require a large number of simulation runs before the system output statistics can be analyzed. As model fidelity increases, sampling techniques become computationally infeasible, and Reliability methods have gained popularity as an analysis method that requires significantly fewer simulation runs. Reliability analysis is an analytic technique which finds a particular point in the design space that can accurately be related to the probability of system failure. However, application to dynamic systems have remained limited. In the following thesis a First Order Reliability Method (FORM) is used to determine the failure probability of a dynamic system due to system/input uncertainties. A pendulum cart system is used as a case study to demonstrate the FORM on a dynamic system. Three failure modes are discussed which correspond to the maximum pendulum angle, the maximum system velocity, and a combined requirement that neither the maximum pendulum angle or

system velocity are exceeded. An explicit formulation is generated from the implicit formulation using a Response Surface Methodology, and the FORM is performed using the explicit estimate. Although the analysis converges with minimal simulation computations, attempts to verify FORM results illuminate current limitations of the methodology. The results of this initial study conclude that, currently, sampling techniques are necessary to verify the FORM results, which restricts the potential applications of the FORM methodology. Suggested future work focuses on result verification without the use of Importance sampling which would allow Reliability methods to have widespread applicability.

Table of Contents

Abstract iv

List of Tables viii

List of Figures ix

Chapter 1. Introduction and Motivation 1

Chapter 2. Reliability Analysis 7

2.1 Introduction 7
2.2 A Brief History of Reliability Analysis 8
2.3 Introduction to Reliability Analysis 10
2.4 First Order Reliability Method 15
2.4.1 Most Probable Point Search Algorithm 18
2.4.2 FORM of a Cantilever Beam 21
2.5 Sensitivity Analysis 23
2.5.1 FORM and Sensitivity of Crack Propagation 24

Chapter 3. Response Surface Generation 26

3.1 Introduction 26
3.2 Response Surface Methodology 27
3.3 Response Surface Application 32
3.3.1 Analytic Design Point 33
3.3.2 RSM Application 35
3.3.3 Comparison with Reference 39
3.4 Response Surface FORM Comparison 40
3.5 Methodology Comments 41

Chapter 4. RS and FORM of a Dynamic Model **43**
4.1 Introduction 43
4.2 Equations of Motion 45
4.3 Limit State Derivation 47
4.4 Sensitivity Analysis 49
4.5 RSM 55
4.6 FORM 59
4.7 Methodology Comments 61

Chapter 5. Conclusions and Future Work **63**

Appendices **66**

Appendix A. Dynamic Analysis **67**
A.1 Introduction 67
A.2 Basic Kinematic Equations 67
A.3 Angular Momentum 70
A.4 Derivation of the Bicycle Model 72
A.5 Pendulum Cart System 75
A.5.1 Pendulum 77
A.5.2 Cart 82
A.5.3 Constraint Equations and Final Equations of Motion . . 83

Appendix B. Python Code **85**
B.1 Introduction 85
B.2 Set Up Code 86
B.3 Simulation Code 87
B.4 RSM Code 90
B.5 FORM Code 98

Bibliography **101**

List of Tables

2.1 Comparison of Python FORM and Analytic FORM 23
2.2 Comparison of Python Sensitivity and Reference Sensitivity . 25

3.1 Analytic vs Estimate in FORM 37
3.2 MPP with Updated Staring Point 38
3.3 RSM Cantilever Beam Summary 41
3.4 RSM Solder Void Growth Summary 41

4.1 Nominal Ensemble Simulation Values 46
4.2 Response Surface Coefficients in Higher Order Space 51
4.3 Sensitivity Results 53
4.4 RSM Coefficient Summary 57
4.5 FORM Results Summary 60

List of Figures

1.1 Integration of Limit State Over a Design Space [3] 2
1.2 Pendulum Cart System 4

2.1 Probability Distribution Function with Performance Function [3] 11
2.2 Probability Distribution Function In U-Space [3] 14
2.3 MPP Search Algorithm Flow Chart [3] 20
2.4 Cantilever Beam Example [3] 21

3.1 RSM Explanation 28
3.2 Exact Design Point in Normalized Space 35
3.3 Normalized RSM Iteration 1 36
3.4 Reference Comparison Discrepancies [6] 39

4.1 Pendulum Cart System 45
4.2 Dynamic Response to a Unit Step Force 47
4.3 Design Point Case 1 52
4.4 Design Point Case 2 53
4.5 Exact Design Point 55
4.6 Case 1 RSM Iterations 57
4.7 Case 2 RSM Iteration 1,2 58
4.8 Case 3 RSM Iterations 58
4.9 Response at First Iteration Expansion Points 61

A.1 A vector viewed in a non-inertial frame 68
A.2 Non-inertial and Inertial Frame with Equal Instantaneous Velocity 69
A.3 Bicycle Model Schematic 73
A.4 Equivalent Bicycle Model Schematic 74
A.5 Cart and Pendulum Ensemble 76
A.6 Pendulum Force Diagram 78

A.7 Cart Force Diagram 83

Chapter 1

Introduction and Motivation

As computational tools advance, designers are able to create higher fidelity models of real world device performance. However, as model complexity increases, care must be taken to account for both model and parameter uncertainties. Importance sampling techniques, for example Monte Carlo and Latin Hypercube, are extremely popular for analyzing complex systems with uncertainties. Sampling techniques perform a large number of simulation runs that are analyzed to compute system statistics. However, the computational expense of high fidelity model simulations can limit the viability of sampling techniques. Even with advances in computing, the computational expense of CFD, FEA, and multibody simulation analysis is generally to large to additionally employ Importance sampling. Unlike Importance sampling, Reliability analysis is an analytic technique that finds a particular point in the design space which can accurately be related to the probability of system failure. The popularity of Reliability analysis has rapidly increased due to fewer required simulation runs, while still considering a large number of uncertain parameters.

In Reliability analysis a limit state function, (g), separates the design space into safe and failure regions. Integrating the joint probability density

function over the safe region of the design space gives the system reliability. Figure 1.1 shows an example of a two variable design space along with a limit state function $g(X_1, X_2)$. The design space represents the uncertain system variables, and the joint probability distribution function, $f(X_1, X_2)$, describes the likelihood that the system realization is given by the pair (X_1, X_2). By convention, $g > 0$ represents a safe region, and $g \leq 0$ represents an unsafe region. Integrating $f(X_1, X_2)$ over the space where $g > 0$ provides the probability that the system realization is in the safe region.

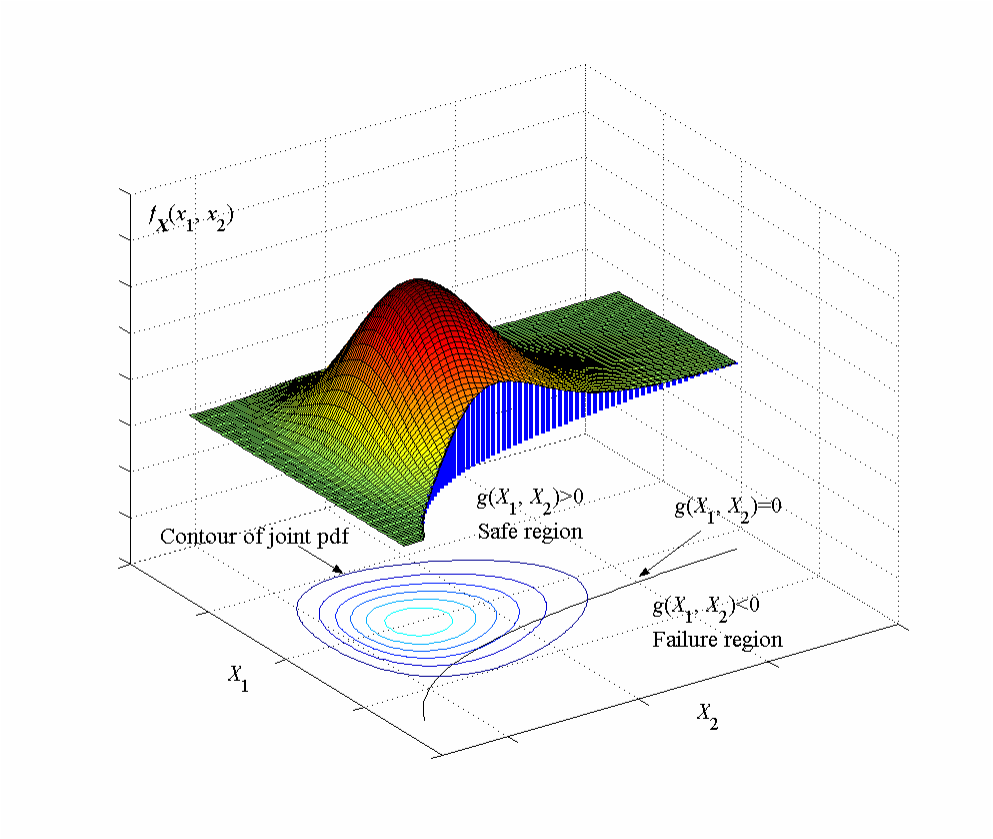


Figure 1.1: Integration of Limit State Over a Design Space [3]

Traditionally, reliability methods have been widely used in structural engineering where finite element analysis is prevalent. Recent work has extended reliability techniques to dynamic system analysis. In Sigbjornsson [5], Reliability analysis was used to analyze a single vehicle accident. The exact conditions at the time of the accident were unknown, and Reliability analysis was used to assess the relative importance of contributing factors. A limit state function specified the maximum allowable side slip at the front and rear tires. A sensitivity analysis was then performed to assess the importance of contributing factors, e.g. wind speed, vehicle speed and turning curvature, in causing the identified accident. Sigbjornsson [5] was then able to provide the most probable cause of the single vehicle accident.

The same techniques used in Sigbjornsson [5] can be used by design engineers to assess model sensitivity to uncertain system parameters. Chen [4] uses a high fidelity vehicle model to analyze the effects of ground excitation and super-elevation in vehicle roll over accidents. However, the problem formulation in Chen [4] requires an augmented form of the reliability methods used in Sigbjornsson [5]. In Sigbjornsson [5] the limit state was an explicit function of the variables of interest, while in Chen [4] the limit state is an implicit function. An implicit limit state occurs because the variables of interest must be determined through a simulation before evaluating the limit state function. To estimate an explicit formulation of the implicit limit state a response surface with be generated. After generating the explicit formulation, the reliability procedure proceeds normally.

The following work demonstrates, explicitly, the First Order Reliability Method (FORM) analysis of a dynamic system. The particular system of interest is a pendulum-cart system, Figure 1.2. Three limit states are considered

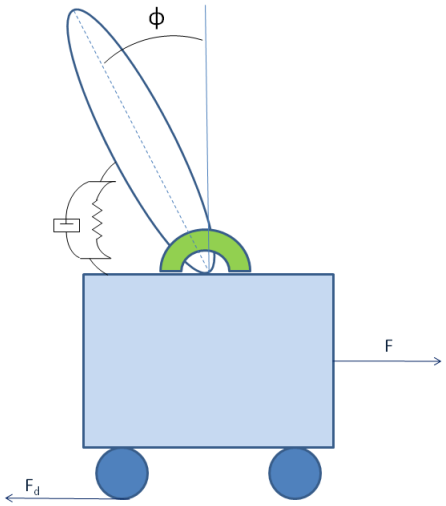


Figure 1.2: Pendulum Cart System

specifying the maximum pendulum angle, the maximum system velocity, and a combined requirement that neither the maximum angle nor the maximum velocity is exceeded.

Because a simulation has to be run to evaluate the limit state variables, the limit states identified are implicit functions. To generate an explicit estimate a response surface technique will be used. A response surface is generated for each of the three limit states, and a FORM is then performed with the estimate. The results of the FORM facilitate a sensitivity analysis which identifies the critical parameters for system safety. Because of the Reliabil-

ity method flexibility, higher fidelity models can supplant the simple models discussed with minimal alteration to the methodology.

Chapter 2 discusses the general Reliability Analysis problem formulation. Section 2.2 provides a brief synopsis of the Reliability Method development to motivate the procedural steps developed in future sections. The general problem formulation is discussed at length in Section 2.3, and the First Order Reliability Method is laid out in detail in Section 2.4. A cantilever beam example, Du [3], is used to illustrate the methods developed in Section 2.4. Section 2.5 concludes by discussing how the FORM results are used to perform a sensitivity analysis, and an example from Robinson [7] is used to demonstrate the methodology.

Chapter 3 introduces the Response Surface Methodology (RSM) which is used to approximate the implicit function with an explicit function. The procedural steps are outlined in Section 3.2, and demonstrated with a reference example from Rajashekhar [6] in Section 3.3. The FORM examples discussed in Chapter 2 are reworked in Section 3.4 using a response surface approximation to show that the estimates provide equivalent FORM results as the analytic formulations. The chapter concludes with important considerations of using the Response Surface generation techniques in Section 3.5.

After outlining the Reliability and Response Surface procedures in Chapter 2 and 3, Chapter 4 applies the methodologies to the pendulum cart system. The equations of motion are discussed in Section 4.2, and nominal simulation parameters are identified. In Section 4.3 the limit state functions

are given quantitative formulations. Each simulation parameter is assumed to have a given uncertainty, and a system sensitivity analysis is performed in Section 4.4. The results of the sensitivity analysis are used to reduce the problem design space from eight variables to two. The RSM and FORM are then performed on the reduced space to demonstrate the methodology application. Section 4.5 generates a response surface for the three limit states identified, and Section 4.6 demonstrates a FORM analysis for each limit state function. The chapter concludes with important methodology considerations in Section 4.7.

Finally, Chapter 5 presents conclusions based on the completed work. Benefits and drawbacks of the Reliability Analysis and Response Surface methods are discussed heuristically, and possible avenues of further exploration are identified.

Chapter 2

Reliability Analysis

2.1 Introduction

Reliability analysis is a useful technique to analyze computationally intense models with a large number of system uncertainties. To motivate the First Order Reliability Method (FORM) derivation, Section 2.2 and 2.3 discuss the background and theory of Reliability analysis. Section 2.2 provides a brief evolution of the Reliability methods to motivate assumptions and simplifications in the general reliability problem. Section 2.3 sets up the reliability problem and shows how advancements discussed in Section 2.2 affect the Reliability analysis procedures. Section 2.3 discusses the problem in sufficient theoretical detail to provide a background for the FORM results derived in Section 2.4. Particular attention is given to the geometric implications of the FORM formulation.

The geometric implications discussed in Section 2.4 lead to the definition of the reliability index, β, and the probability of failure is then related to the reliability index. As will be shown, the reliability index is sensitive to the expansion point of the limit state function, and Section 2.4.1 discusses an iterative method to find the expansion point. An example of the methodology is

shown in Section 2.4.2 to demonstrate the FORM with a cantilever beam problem. Section 2.5 demonstrates how the FORM results can be used to determine the system sensitivity to parameter variation. A set of Python programs, Appendix B, were created to perform the iterative procedures identified, and the program results are compared to the analytic results as validation.

2.2 A Brief History of Reliability Analysis

To motivate the major assumptions made during a Reliability analysis, it is important to understand how Reliability methods evolved to address specific issues. Initially, reliability analysis focused on Mean Value Methods, such as Mean Value First Order-Second Moment (MVFOSM) methods. In MVFOSM methods the Taylor Series expansion of the performance function, $g(X)$, is centered at the mean of the random variables of interest. The integration over the safe probability space, $g > 0$, provides the system reliability. However, a major issue with MVFOSM methods are that solutions are not algebraically invariant Robinson [7]. The lack of invariance implies that the solution is dependent on the specific algebraic formulation, i.e $\frac{a}{b} = 1$ could yield a different solution then $a - b = 0$.

To deal with the lack of invariance in the MVFOSM methods Hasofer-Lind proposed an orthogonal transformation of the random variables of interest and a specific expansion center for the limit state in the transformed space. Applying the orthogonal transformation creates a set of independent random variables which, as will be shown, greatly simplifies the integration over the

probability space of interest.

The expansion point, known as the Most Probable point (MPP), is the point lying on the zero contour of the limit state which maximizes the joint probability density function. At the zero contour of the limit state function, the system is passing from a safe to an unsafe region. By maximizing the joint probability density function along the zero contour, the most likely point of system failure can be identified. Using the MPP as the limit state expansion center creates a locally accurate approximation of the system failure boundary in the region of highest probability density.

While the methodology laid out by Hasofer-Lind solved the invariance issues and greatly simplified the design space integration, the methodology was inaccurate when non-normal random variables are considered. The Hasofer-Lind method approximates the random variable of interest as a Gaussian random variable with equal first two moments. The standard normal transformation is then used to transform the Gaussian random variable into a standard Gaussian random variable. However, if the underlying probability density function is highly non-Gaussian, then the estimation may have gross errors Robinson [7]. For example, approximating a Weibull distribution as a Gaussian with equal first two moments will be extremely inaccurate at all but a few locations.

Rackwitz and Fiessler augmented the Reliability methodology further by incorporating detailed statistical information of the underlying design variables into the analysis. Using the work of Rosenblatt, who developed a unique

transformation from non-Gaussian to Gaussian random variables, Rackwitz and Fiessler augmented the Reliability methods to allow analysis of non-Gaussian random variables. The Rosenblatt transformation is a general transformation which equates the cumulative probability functions of two random variables, one of which is a standard random variable, at a given point Robinson [7]. Rackwitz and Fiessler used the Rosenblatt transformation to augment the methodology of Hasofer-Lind and improve the analysis for non-Gaussian random variables.

The current section provided a brief history of Reliability methods to provide high level motivation for the simplifying assumptions made in future sections. Section 2.3 will look at the general Reliability problem in more rigorous detail to provide the reader with the necessary background and motivation for the First Order Reliability Method derivation discussed in Section 2.4.

2.3 Introduction to Reliability Analysis

In Section 2.2 a brief evolution of Reliability Methods was provided to motivate the mathematical assumptions made throughout the methodology derivation. Unlike Importance sampling, Reliability analysis is an analytic technique which finds a particular point in the design space that can be accurately related to the probability of system failure. The following section provides a detailed discussion on the Reliability problem to provide background and motivation for the First Order Reliability Method (FORM) discussed in Section 2.4.

First, consider a scalar function based on a number of random variables, $g(X_1, \ldots, X_n)$. The limit state function, g, separates the design space into safe and failure regions. By convention, g is greater than zero in a safe region and less than zero in a failure region. For example, consider a two dimensional design space (X_1, X_2) with a probability density function, $f_x(X_1, X_2)$, and limit state function, $g(X_1, X_2)$, shown in Figure 2.1. The design space represents

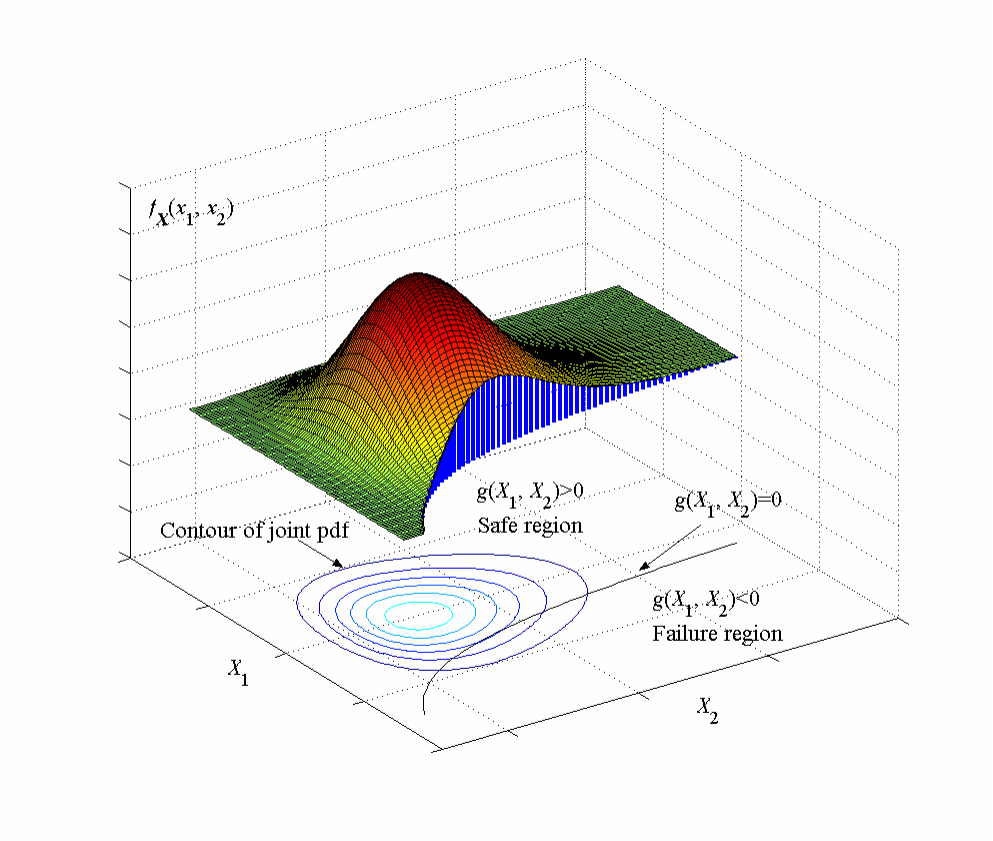


Figure 2.1: Probability Distribution Function with Performance Function [3]

the uncertain system variables, and the joint probability density function, $f_x(X_1, X_2)$, describes the likelihood that the system realization is given by the

pair (X_1, X_2). The the probability of system failure is found by integrating $f_x(\mathbf{X})$ over the region where $g(\mathbf{X}) \leq 0$

$$p_f = P\{g(\mathbf{X}) < 0\} = \int_{g(\mathbf{X})<0} f_x(\mathbf{X})\mathrm{d}x \tag{2.3.1}$$

and the reliability is simply

$$R = 1 - p_f = P\{g(\mathbf{X}) > 0\} = \int_{g(\mathbf{X})>0} f_x(\mathbf{X})\mathrm{d}x \tag{2.3.2}$$

The probability integration in 2.3.1 and 2.3.2 is the volume under the joint probability density function in the safe or failure regions, respectively. However, direct evaluation of the integral is computationally infeasible for most engineering applications.

Generally, the number of random variables under consideration is large and the multidimensional joint probability integration does not have a closed form. Additionally, both the joint probability density function, $f_x(\mathbf{X})$, and the integration boundary, $g(\mathbf{X})$, can be nonlinear and multidimensional. Further complicating issues, many engineering applications do not have an explicit formulation for the performance function, $g(\mathbf{X})$, and instead the variables of interest are obtained from a simulated model, e.g. finite element, computational fluid analysis, dynamic simulation, etc., Du [3].

Because of the issues directly evaluating the integrals of 2.3.1 and 2.3.2, the first step in Reliability Analysis is to simplify the integrand. To simplify the joint probability density function the original random variables $X_1, \ldots, X_n$ are transformed into a standard normal space. Notationally, the original space will

be denoted as X-space and the transformed space as U-space. The Rosenblatt transformation maps X-space to U-space by requiring the cumulative density function of both random variables be equal at the evaluation point

$$F_{X_i}(x_i) = \Phi(u_i) \tag{2.3.3}$$

where $\Phi(u_i)$ is the cumulative distribution function of the standard normal distribution. The transformation from X-space to U-space also changes the performance function from $g(\mathbf{X})$ to $g(\mathbf{U})$. The performance function transformation is generally non-linear and can be complicated since

$$X_i = F_{X_i}^{-1}[\Phi(U_i)] \tag{2.3.4}$$

giving a performance function

$$g(\mathbf{X}) = g(F_{\mathbf{X}}^{-1}[\Phi(\mathbf{U})] \tag{2.3.5}$$

Despite the transformation, the safe region and failure regions are still given by $g > 0$ and $g \leq 0$, respectively. The probability integration, 2.3.1, then becomes

$$p_f = P\{g(\mathbf{U}) < 0\} = \int_{g(\mathbf{U})<0} \phi(u)\mathrm{d}u \tag{2.3.6}$$

where $\phi(u)$ is the joint probability density function in U-space. Since the transformation to U-space is an orthogonal transformation, the random variables become independent with a standard normal distribution. The integration, 2.3.6, is then simplified to

$$p_f = \int \dots \int \prod \frac{1}{\sqrt{2\pi}} \exp(\frac{-1}{2} u_i^2)\mathrm{d}u_1 \dots \mathrm{d}u_n \tag{2.3.7}$$

The probability density function contours of the two dimensional problem discussed earlier, after transformation into the U-space, is shown in Figure 2.2. Here the integration has been greatly simplified by the given transformation.

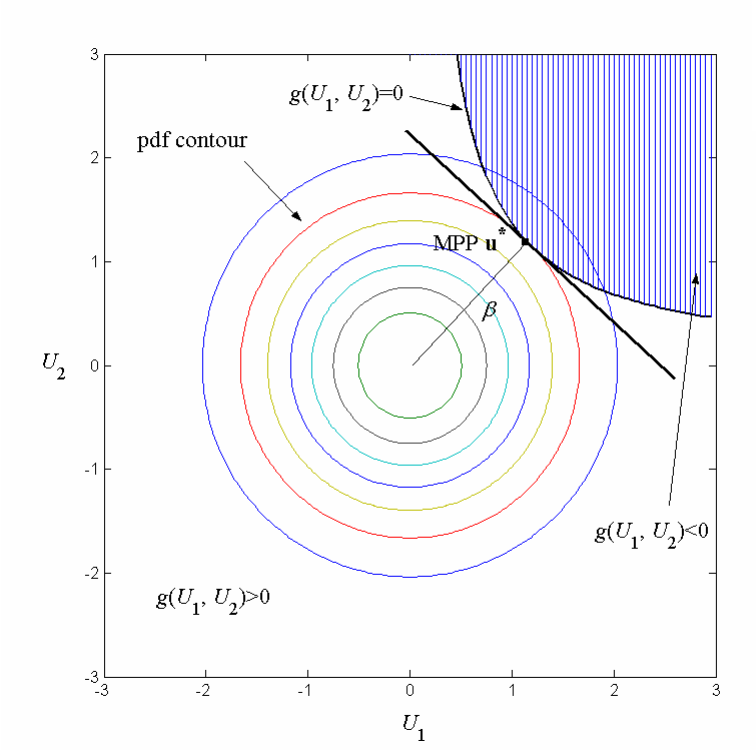


Figure 2.2: Probability Distribution Function In U-Space [3]

In the following work, the random variables will be assumed to have a Gaussian probability density function. Therefore the complications of the Rosenblatt transformation can be largely ignored. The orthogonal transformation of a Gaussian random variable to a standard Gaussian random variable is given by

$$U = \frac{X - \mu_X}{\sigma_X} \tag{2.3.8}$$

where U is the standard normal random variable, X is the general Gaussian random variable, μ_X is the mean of X, and σ_X is the standard deviation of X. For any Gaussian random variable, there is no accuracy lost with the transformation given by 2.3.8.

To simplify the integration boundary a First or Second order approximation of the limit state is constructed Du [3]. The First Order Reliability Method (FORM) and the Second Order Reliability Method (SORM) refer to the approximation order of the limit state, and are two common methods used in engineering analysis. Section 2.4 discusses the formulation of the FORM in detail and demonstrates the method application on several sample problems.

2.4 First Order Reliability Method

As discussed at the conclusion of Section 2.3, the First Order Reliability Method (FORM) refers to the order of the limit state approximation used to simplify the integration boundary of 2.3.1. The First Order Reliability Method uses a first order Taylor series approximation of the performance function

$$g(U) \approx L(U) = g(u^*) + \nabla g(u^*)(U - u^*) \tag{2.4.1}$$

where ∇ represents a gradient operator. Recall that the transformation from X-space to U-space, using the standard normal transformation also transforms the limit state function. U is the standard normal random variable, zero mean and unit variance.

Choosing the expansion point on the limit curve $g(u^*) = 0$ simplifies

the Taylor series expansion to

$$g(U) \approx L(U) = \nabla g(u^*)(U - u^*) \tag{2.4.2}$$

Noticing that $L(U)$ is a linear function of standard random variables implies that $L(U)$ is a normally distributed random variable with mean and standard deviation given by

$$\mu_L = E[\nabla g(u^*)(U - u^*)] = -\nabla g(u^*)u^* = -\sum_i \frac{\partial g(U)}{\partial U_i}\bigg|_{u^*} u_i^* \tag{2.4.3}$$

and

$$\sigma_L^2 = E[[\nabla g(u^*)(U - u^*) - \mu_L][\nabla g(u^*)(U - u^*) - \mu_L]^T] \tag{2.4.4}$$

which reduces to

$$\sum_i (\frac{\partial g(U)}{\partial U_i}\bigg|_{u^*})^2 \tag{2.4.5}$$

making the failure probability

$$p_f \approx P\{L(U) < 0\} = \Phi(\frac{-\mu_L}{\sigma_L}) \tag{2.4.6}$$

Relation of 2.4.6 to geometric properties simplifies the analysis Du [3]. Recall that Hasofer-Lind, Section 2.2, proposed the Most Probable Point (MPP) as the expansion point to deal with problem invariance. Following the work of Hasofer-Lind, the MPP minimizes the norm, defined as β, of a vector from the origin to the limit state zero contour. The MPP now defines the radius of a great circle, and by definition, the gradient points outwards from lines of

constant value. Therefore the gradient of $g(U)$ is in a direction opposite to β. Normalizing u^* and equating to the gradient of $g(U)$ yields

$$\frac{u^*}{||u^*||} = \frac{u^*}{\beta} = -\frac{\nabla g(U)}{||\nabla g(U)||} \tag{2.4.7}$$

and

$$u^* = -\beta \frac{\nabla g(U)}{||\nabla g(U)||} \equiv -\beta \mathbf{a} \tag{2.4.8}$$

Recalling 2.4.6 and plugging in values for μ_L and σ_L yields

$$p_f \approx \Phi(\mathbf{a}u^{*T}) \tag{2.4.9}$$

where au^{*T} is the inner product of a unit vector and the vector from the origin to the MPP. Plugging in 2.4.8 into 2.4.9 yields

$$p_f \approx P\{L(U) < 0\} = \Phi(\mathbf{a}u^{*T}) = \Phi(-\beta \mathbf{a}\mathbf{a}^T) = \Phi(-\beta) = 1 - \Phi(\beta) \tag{2.4.10}$$

and the reliability is given by

$$R = 1 - p_f = \Phi(\beta) \tag{2.4.11}$$

Because Reliability methods are extremely sensitive to the limit state expansion point, specific discussion needs to be given to finding the MPP. One iterative search algorithm was shown by Du [3], which is discussed further in Section 2.4.1. The search algorithm is not necessarily optimal, but shows good convergence speed and robustness in general problem applications.

2.4.1 Most Probable Point Search Algorithm

The algebraic invariance of the First Order Reliability Methods (FORM) is dependent the expansion point of the limit state function. The MPP point, by definition, minimizes the distance from the origin to the zero contour of limit state function. Solving for the MPP is a nonlinear constrained minimization problem, with the objective of maximizing the joint probability density function, $f_u(\mathbf{U})$, subject to the constraint that $g(U) = 0$. Maximizing the joint probability density function is equivalent to minimizing $\sum u_i^2$ since

$$\prod_i \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}u_i^2) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}\sum_i u_i^2) \tag{2.4.12}$$

Following the methods discussed in Du [3], an iterative search algorithm will be used to determine the MPP in U space.

Consider the first order Taylor Series expansion of the limit state function

$$g(u) = g(u^k) + \nabla g(u^k)(u - u^k)^T \tag{2.4.13}$$

which implies the next iteration, u^{k+1} will lie on the line

$$g(u^{k+1}) = g(u^k) + \nabla g(u^k)(u^{k+1} - u^k)^T \tag{2.4.14}$$

since the contour of interest is $g(u) = 0$ a requirement that $g(u^{k+1}) = 0$ can be imposed making 2.4.14

$$0 = g(u^k) + \nabla g(u^k)(u^{k+1} - u^k)^T \tag{2.4.15}$$

and using the geometric properties discussed in Section 2.4

$$u^k = -\beta^k a^k \tag{2.4.16}$$

Because u^{k+1} and a^k are perpendicular to 2.4.14

$$u^{k+1} = -\beta^{k+1} a^k \tag{2.4.17}$$

Plugging in 2.4.17 to 2.4.14 and rearranging yields the recursive equations for the new beta

$$\beta^{k+1} = \beta^k + \frac{g(u^k)}{||\nabla g(u^k)||} \tag{2.4.18}$$

and the update point is given by

$$u^{k+1} = -a^k \{\beta^k + \frac{g(u^k)}{||\nabla g(u^k)||}\} \tag{2.4.19}$$

To use the recursive formulas, a starting point u^0 is required. Generally, the origin is used as the starting point. Three convergence criteria may be used to terminate the MPP search progress

1. If $||u^{k+1} - u^k|| \leq \epsilon_1$ stop
2. If $||\nabla g(u^{k+1}) - \nabla g(u^k)|| \leq \epsilon_2$ stop
3. If If $||\beta^{k+1} - \beta^k|| \leq \epsilon_3$ stop

The recursive equations were then programmed into Python, Appendix B.5 following the flow chart in Figure 2.3. At the initial starting point the limit state partial derivative, $\nabla g(u)$ and the magnitude of the partial derivative, $||\nabla g(u)||$, are calculated. Additionally, the limit state, $g(u)$ is evaluated. **a** is then determined as well as β_{new} using the limit state and partial limit state

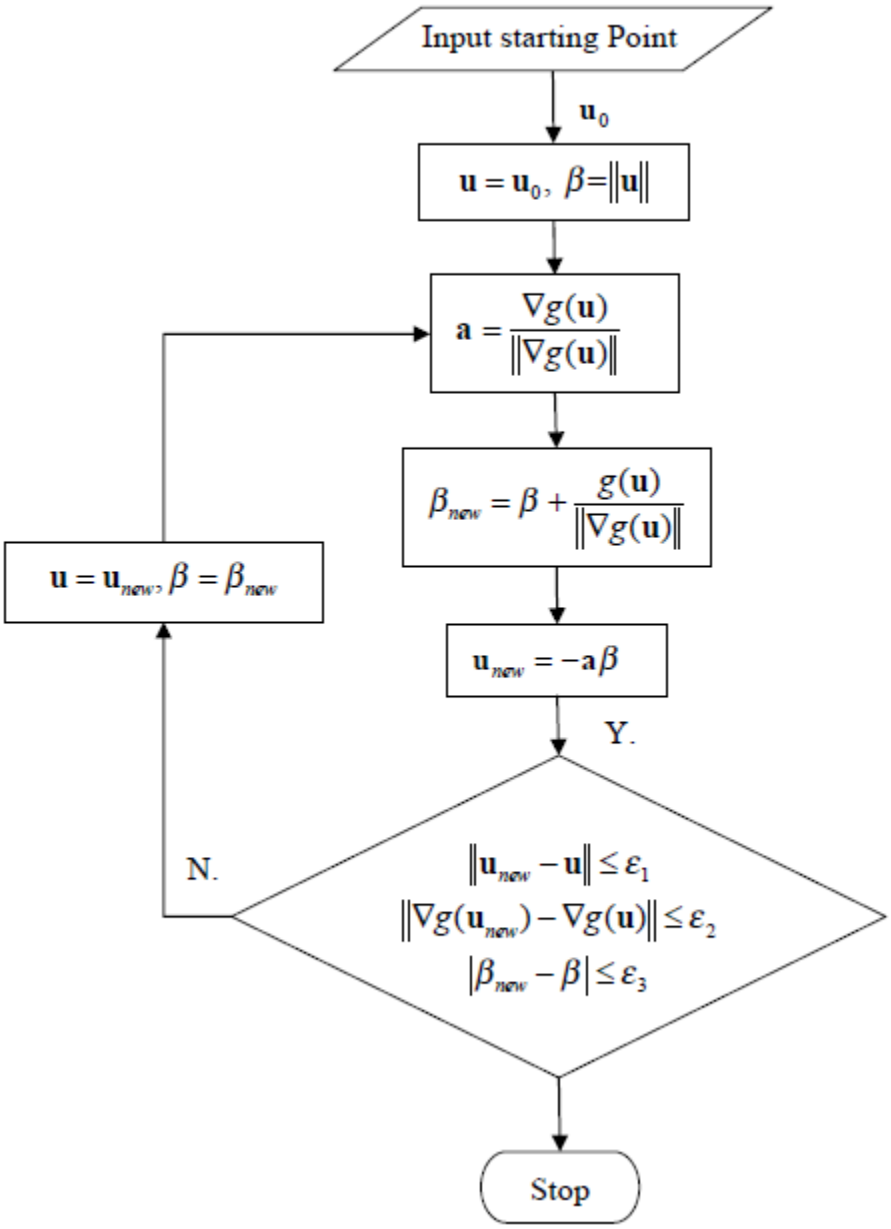


Figure 2.3: MPP Search Algorithm Flow Chart [3]

evaluations. The next point in U-space, u_{new} is calculated, and the convergence criteria evaluated.

The search algorithm discussed was chosen due to rapid convergence speed and ease of implementation. However, it may fail to converge for some problems, for example, oscillating between two points with equal likelihood Du [3]. The oscillation issue will be encountered in Section 3.3 when using an approximation of the analytic limit function. In the following Section 2.4.2 the FORM will be applied to a static cantilever beam to demonstrate the methodological procedures.

2.4.2 FORM of a Cantilever Beam

In Du [3] a cantilever beam, Figure 2.4, is used to demonstrate the FORM procedures. System failure is said to occur when the maximum tip deflection exceeds a critical value, D_0. The performance function is given as maximum deflection minus the current tip deflection as

$$g = D_0 - \frac{4L^3}{Ewt}\sqrt{(\frac{\mathbf{P_y}}{t^2})^2 + (\frac{\mathbf{P_x}}{w^2})^2} \tag{2.4.20}$$

where the maximum tip deflection is three inches, $D_0 = 3''$,the modulus of

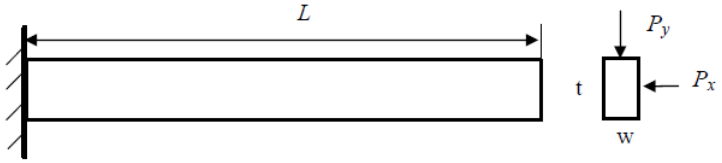


Figure 2.4: Cantilever Beam Example [3]

elasticity is 30×10^6, $E = 30 \times 10^6$psi, the length is 100 inches, $L = 100''$, w and t are the width and height of the cross section, respectively, and are given by $w = 2''$ and $t = 4''$. $\mathbf{P_x}$ and $\mathbf{P_y}$ are the external loads acting on the cantilever beam. The probability distribution of $\mathbf{P_x}$ and $\mathbf{P_y}$ are given by normal distributions $\mathbf{P_x} \sim \mathcal{N}(500, 100)$lb and $\mathbf{P_y} \sim \mathcal{N}(1000, 100)$lb. To transform the normally distributed variables into standard normal variables the following relation is used

$$X = (\mathbf{P_x}, \mathbf{P_y}) = (\mu_{P_x} + \sigma_{P_x}\mathbf{U_x}, \mu_{P_y} + \sigma_{P_y}\mathbf{U_y}) \tag{2.4.21}$$

making 2.4.20

$$g = D_0 - \frac{4L^3}{Ewt}\sqrt{(\frac{\mu_{P_x} + \sigma_{P_x}\mathbf{U_x}}{t^2})^2 + (\frac{\mu_{P_y} + \sigma_{P_y}\mathbf{U_y}}{w^2})^2} \tag{2.4.22}$$

The Python program identified in Appendix B.5, takes a symbolic representation of the limit state function. The value of g is evaluated at the desired point in U-space by 2.4.22. To determine ∇g the limit state function, g, is symbolically differentiated and evaluated at the appropriate point. To demonstrate the Python code's accuracy, the output of the program was compared to the analytical results shown in Du [3]. The comparison is shown in Table 2.4.2, and the program results line up as expected.

The current section demonstrated the FORM for a cantilever beam. The work of Robinson [7] shows how the results of the FORM readily facilitate a sensitivity analysis of the system, and Section 2.5 will discuss the procedure in further detail.

Table 2.1: Comparison of Python FORM and Analytic FORM

	FORM Analytic		FORM Python	
Iteration	U Update	Beta	U	Beta
0	(1.7722, 0.22152)	0	(1.7722, 0.2215)	0.0
1	(1.7375, 0.16391)	1.7859	(1.7375, 0.16391)	1.7859
2	(1.7367, 0.16375)	1.7453	(1.7367, 0.16375)	1.7453
3	(1.7367, 0.16376)	1.7444	(1.7367, 0.163764)	1.7444

2.5 Sensitivity Analysis

Sensitivity analysis quantifies the effect of variable variation on the system output. Focusing on the parameters which have the greatest impact on system performance promotes efficient and tractable systems testing. In the context of analytic methods, sensitivity measures are referred to as importance factors and are defined as

$$\gamma = \left| \frac{-u_i^*}{\beta} \right| = \left| \frac{(\frac{\partial G(u)}{\partial u_i})\Big|_{u*}}{\sum (\frac{\partial G(u)}{\partial u_i})^2\Big|_{u*}} \right| \tag{2.5.1}$$

Geometrically, 2.5.1 represents the directional cosine of the random variable in the reduced design space, and two computational checks are given by

$$\sum_i \gamma_i^2 = 1 \tag{2.5.2}$$

and

$$-1 \leq \gamma \leq 1 \tag{2.5.3}$$

The magnitude of γ represents the random variable's effect on the safety index, and therefore the variable's impact on the overall system reliability. If the system variable has minimal influence on the overall system reliability, then those variables can be reasonably omitted from the analysis Robinson [7]. Section

2.5.1 demonstrates the sensitivity calculations with an example describing the void growth in solder joints.

2.5.1 FORM and Sensitivity of Crack Propagation

To demonstrate the sensitivity analysis discussed, an example, Robinson [7], will be reworked. The problem models void growth within a solder interconnect. The crack size, A, is a function of the initial stress, ν_0 and the grain size, L, as

$$A = \nu_0[.01694 - .01353 \exp(-\frac{.4158}{L})] \tag{2.5.4}$$

where the mean and coefficient of variation of ν_0 and and L are $(300, .2)$ and $(1.25, .1)$, respectively. The maximum crack length, Ac, is 2.7 microns and the corresponding limit state function is

$$g = 2.7 - \nu_0[.01694 - .01353 \exp(-\frac{.4158}{L})] \tag{2.5.5}$$

Using the Python code, Appendix B, a few lines must be altered to analyze the given problem. First, in Set_Up.py the mean and coefficient of variation for the variables need to be changed appropriately. The limit state function in FORM.py needs to be changed to reflect 2.5.5. Finally, the final MPP value needs to be divided by the reliability index β to return the sensitivity of each variable.

The results of the Python program are compared to the reference in Table 2.5.1. The FORM and sensitivity results from the Python program generate equivalent results to those seen in Robinson [7].

Table 2.2: Comparison of Python Sensitivity and Reference Sensitivity

Python Program			Reference Program		
U Space	Beta	Sensitivity	U Space	Beta	Sensitivity
(1.127, -0.32)	1.17	(0.9619, -0.273)	(1.127, -0.32)	1.17	(0.962, -0.273)

Chapter 3

Response Surface Generation

3.1 Introduction

In the previous chapter, the First Order Reliability Method (FORM) was derived and demonstrated for a cantilever beam with uncertain force inputs. An iterative method was used to determine the Most Probable Point (MPP),and the reliability parameter, β, was then shown to be norm of the vector from the origin, in U-space, to the MPP. The probability of failure was shown to be $\Phi(\beta)$, where $\Phi()$ is the cumulative distribution function. A sensitivity analysis was also performed, using the results of the FORM, and the effect of parameter variation on system reliability was quantified.

The analysis techniques discussed in Chapter 2 use the partial derivative of the limit state function to evaluate the reliability, sensitivity, and successive iteration parameters. To calculate the partial derivative, an explicit formulation is necessary. Without an explicit function, the Taylor Series expansion can not be calculated, and the FORM can not be carried out. However, most dynamic systems have implicit limit state functions where simulations must be run before the limit state evaluation. The following chapter discusses a methodology for approximating an implicit function with an explicit function.

Section 3.2 follows the work of Rajashekhar [6] to generate a response surface approximation to an implicit limit state. The response surface is assumed to have a quadratic structure. By evaluating the limit state function at a set of experimental points, specified in Rajashekhar [6], the quadratic coefficient values can be found by solving the resulting algebraic equations. The expansion center is then moved and the iterative process continues until a convergence criteria, δ, is satisfied. An iterative method with convergence criteria in Rajashekhar [6] is an improvement on the work done in Bucher [2] to monitor the approximation accuracy near the design point. Section 3.3 reworks the two parameter limit state function from Rajashekhar [6] to illustrate the response surface generation methodology. Discrepancies and possible resolutions with the published results are identified and discussed as appropriate. Due to discrepancies identified with the example in Rajashekhar [6], Section 3.4 reworks the cantilever beam and solder void growth examples from Chapter 2 using the RSM. The examples are used to further verify agreement of the FORM results between the response surface estimate and the analytic results. Finally, Section 3.5 concludes with a discussion of methodology considerations.

3.2 Response Surface Methodology

The methodology laid out in Rahashekhar [6] will be used to generate an explicit response surface estimate of an implicit limit state function. It is worthwhile to first discuss the Response Surface Methodology (RSM) heuristically to motivate the mathematical derivation. Consider Figure 3.1, discussed

further in Section 3.3.2 with relation to the problem proposed in Section 3.3. The figure shows the zero contour of the analytic limit state, and the analytic

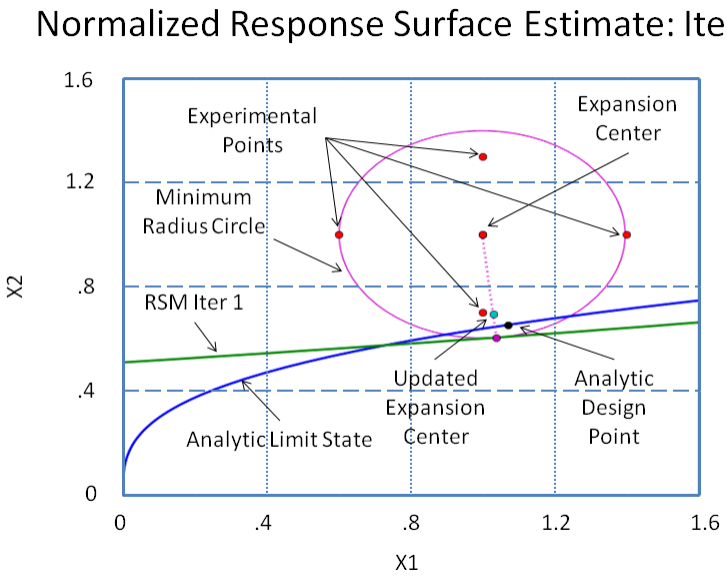


Figure 3.1: RSM Explanation

design point for the limit state. To begin the estimation procedure, an initial expansion point is required. For the procedure discussed, the variable means, $(\mu_{X1} = 1.0, \mu_{X2} = 1.0)$, will be used as the starting point. Four experimental points at $(\mu_{X1} \pm c, \mu_{X2})$ and $(\mu_{X1}, \mu_{X2} \pm c)$ are then generated. At the mean and four additional points the actual limit state is evaluated. From the limit state evaluations, the coefficients of a response surface estimate are determined. The zero contour of the response surface is shown alongside the analytic zero contour in Figure 3.1. The point which lies along the estimate zero contour which minimizes the distance to the starting point is determined. A linear in-

terpolation, with weightings given by analytic limit state evaluations, is then performed to determine the subsequent expansion center. The method is then repeated until a convergence criteria is met. Using the heuristic description as motivation, the RSM will be examined in further detail following Rahashekhar [6].

The explicit limit state will be assumed to be quadratic in random variables X_i and have coefficients a, b_i, and c_i. The form of the limit state estimate is given by

$$g_{est} = a + \sum_{i}^{n} b_i X_i + \sum_{i}^{n} c_i X_i^2 \tag{3.2.1}$$

A quadratic estimate will only be accurate within small variations of the expansion point. To determine the best expansion point, an iterative process is developed.

The iterative method begins by determining the coefficient values of g_{est}. To determine the coefficient values, a set of test points are generated, deterministically, by populating the following matrix

$$A = \begin{bmatrix} \mu_{X_1} & \mu_{X_2} & \cdots & \mu_{X_n} \\ \mu_{X_1} + h\sigma_{X_1} & \mu_{X_2} & \cdots & \mu_{X_n} \\ \mu_{X_1} - h\sigma_{X_1} & \mu_{X_2} & \cdots & \mu_{X_n} \\ \mu_{X_1} & \mu_{X_2} + h\sigma_{X_2} & \cdots & \mu_{X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{X_1} & \mu_{X_2} & \cdots & \mu_{X_n} - h\sigma_{X_n} \end{bmatrix} \tag{3.2.2}$$

where μ_{X_i} is the random variable mean, σ_{X_i} is the standard deviation, and h is an arbitrary factor, whose value assignment is discussed later in the section.

The actual limit state, g, is evaluated at each test point, given by the rows of matrix A.

Let B be an $n \times 1$ vector which contains the limit state evaluations at the experimental points given in 3.2.2,

$$B = \begin{bmatrix} g(A[1,1], \ldots, A[1,n]) \\ \vdots \\ g(A[n,1], \ldots, A[n,n]) \end{bmatrix} \tag{3.2.3}$$

C be an $(2n+1)\times 1$ vector containing the quadratic response surface coefficients of 3.2.1,

$$C = \begin{bmatrix} a, b_1, \cdots, b_n, c_1, \cdots, c_n \end{bmatrix}^T \tag{3.2.4}$$

and D be an $n \times (2n+1)$ matrix defined as

$$D = \begin{bmatrix} 1 & \mu_{X_1} & \cdots & \mu_{X_n} & (\mu_{X_1})^2 & \cdots & (\mu_{X_n})^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \mu_{X_1} & \cdots & \mu_{X_n} - h\sigma_{X_n} & (\mu_{X_1})^2 & \cdots & (\mu_{X_n} - h\sigma_{X_n})^2 \end{bmatrix} \tag{3.2.5}$$

The coefficients of g_{est} can then be found by

$$C = D^{-1}B \tag{3.2.6}$$

Using calculated coefficients, an explicit expression for $g_{est}(\mathbf{X})$ is determined Rajashekhar [6]. The arbitrary multiplicative factor, h, describes the expansion width of the response surface estimate, i.e. how many standard deviations away from the expansion center the experimental points should be placed. Experience has shown, Rajashekhar [6], that the arbitrary constant, h, should be set equal to 2.0 until a convergence criteria δ gets sufficiently small $\approx 1e-3$, and then set equal to 1.0 for subsequent iterations.

Referring back to Figure 3.1, the test point matrix specifies the experimental points at which the actual limit state function g is evaluated. The expansion center is denoted as X_m. For the first iteration $X_m = X_\mu$. The goal is then to update the expansion center from X_m to X'_m which provides a closer approximation of the analytic design point. To determine X'_m a point, X_d, is calculated from g_{est} which minimizes the norm $||X_d - X_m||$ subject to the constraint that $g_{est} = 0$. There are several ways to solve a constrained minimization problem. The following work uses a simple loop to determine the minimization point or points. In both Python and Matlab there are built in functions that generate the contours of higher dimension, scalar, functions. The variable values that generate the zero contour can be pulled directly from these built in functions. Because a discrete number of values are returned, a loop can iterate over the returned values and calculate the norm, distance between the point and X_m, at each point. The point which minimizes the norm along the zero contour is X_d.

After computing X_d, a linear interpolation calculates the new design point, X'_m, which lies on the line connecting X_d and X_m such that $g(X'_m) = 0$. The interpolation is given as

$$X'_m = X_m + (X_d - X_m)\frac{g(X_m)}{g(X_m) - g(X_d)} \tag{3.2.7}$$

Setting $X_m = X'_m$ the iterative procedure is repeated. A convergence criteria δ was introduced in Rajashekar [6] which evaluates the norm $||X_d - X_m||$. When $\delta < \epsilon$ the method is considered to have converged.

If the random variables X_i are not independent then cross terms can be considered using the following form

$$g_{est} = a + \sum_i b_i X_i + \sum_i \sum_{j \geq i} c_{ij} X_i X_j \tag{3.2.8}$$

Rajashekhar [6] showed that the addition of cross terms do not generally affect the response surface generation, and cross terms will be ignored in the following work.

The Response Surface Methodology of Rajashekhar [6] will be used in the remainder of this work to generate a response surface approximation to an implicit limit state function. Appendix B.4 contains the Python code used to implement the RSM discussed. Section 3.3 will demonstrate the application of the RSM on a two parameter limit state function.

3.3 Response Surface Application

Following the work of Rajashekhar [6], the response surface techniques are used to create an approximation to an implicit limit state function. Consider a cantilever beam with a rectangular cross section subjected to a uniformly distributed load. The limit state is given by the maximum tip deflection in Rajashekhar [6] as

$$g = -\frac{(w \times b) l^4}{8EI} + \frac{l}{325} \tag{3.3.1}$$

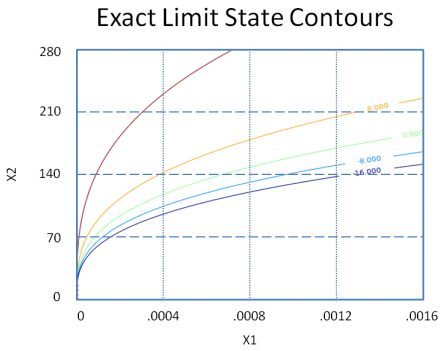
which can be represented numerically as

$$g = 18.46154 - 7.476923 \times 10^{10} \frac{X_1}{(X_2)^3} = 0 \tag{3.3.2}$$

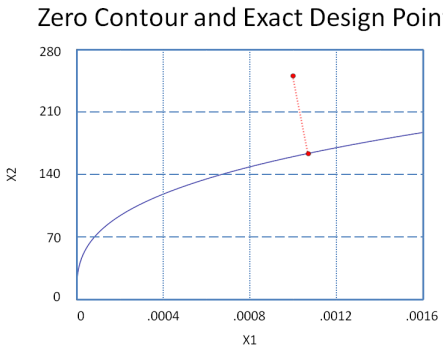
where $\mu_{X_1} = 1e-3\text{N/mm}^2$, $\mu_{X_2} = 250\text{mm}^2$, and the coefficient of variation is given as $V1 = .2$ and $V2 = .15$, respectively. The Python code given in Appendix B.4 is used to perform the iterative RSM, and the FORM results of both the analytic and response surface estimate are compared to verify the methodology.

3.3.1 Analytic Design Point

The first step in the verification of the proposed method is to calculate the analytic design point of the limit state function. To determine the exact design point, a linearly spaced mesh of points were created using the built in Python subroutine meshgrid(). The exact limit state function, equation 3.3.2, was evaluated at every point in the space. A contour plot was then generated using the contour() function. The results of the contour generation can be seen in Figure 3.2(a). The analytic design point is given by the minimum



(a) Contours of Exact Limit State



(b) Exact Design Point

distance between the mean of the random variables and the zero contour of the limit state function. The subroutine get_paths() can be used to extract

the (X_1, X_2) values which correspond the the zero contour of the limit state. Figure 3.2(b) shows the zero contour of the exact limit state, and a heuristic identification of the exact design point can be performed. The intersection of a minimum radius circle, centered at the variable means, and the zero contour is the analytic design point.

To calculate the minimum sphere radius, a loop calculated the Euclidean norm between the current point and the mean along the $g = 0$ contour. The (X_1, X_2) point corresponding to the minimum norm value was then stored as the design point. In the two dimensional case the norm can be written as

$$||\mathbf{X} - \mathbf{X}_\mu|| = \sqrt{(X_1 - X_{\mu 1})^2 + (X_2 - X_{\mu 2})^2} \tag{3.3.3}$$

However, in the example, X_1 is several order of magnitudes smaller than X_2. Hence, the norm is more heavily influenced by changes in X_2 than changes in X_1. To account for the order of magnitude differences, the minimum norm point has to be calculated in a normalized space. Normalizing the design space yields the exact limit contour seen in Figure 3.2. The design point can then be calculated in the normalized space by finding the minimum norm value, and storing the associated design variables.

An analytic design point of $(1.071, 0.6523)$ corresponds to the minimum in the normalized space, and, converting back to the standard space, yields design variable values of $(1.071 \times 10^{-}3, 163)$ in the standard space. Referring back to Figure 3.2(b) the result agrees with the heuristic expectation.

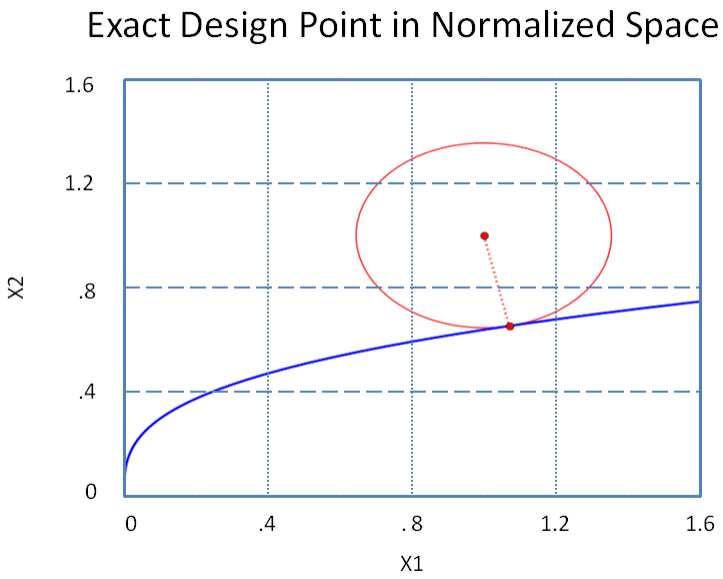


Figure 3.2: Exact Design Point in Normalized Space

3.3.2 RSM Application

To create a quadratic estimate to 3.3.2, the RSM discussed in Section 3.2 will be followed. First, the matrix of design points, 3.2.2, is populated. The results of the first iteration of the RSM are shown in Figure 3.3. The exact limit state function is evaluated at the experimental points, and a response surface estimate is generated solving the resulting algebraic equations, 3.2.6. The minimum radius circle which intersects the limit state estimate is determined by calculating the norm at each point along the contour $g_{est} = 0$, and a linear interpolation, 3.2.7, is used to determine the new expansion center. The convergence criteria δ is evaluated as the distance between consecutive design points, i.e. distance between original and updated expansion centers, and the

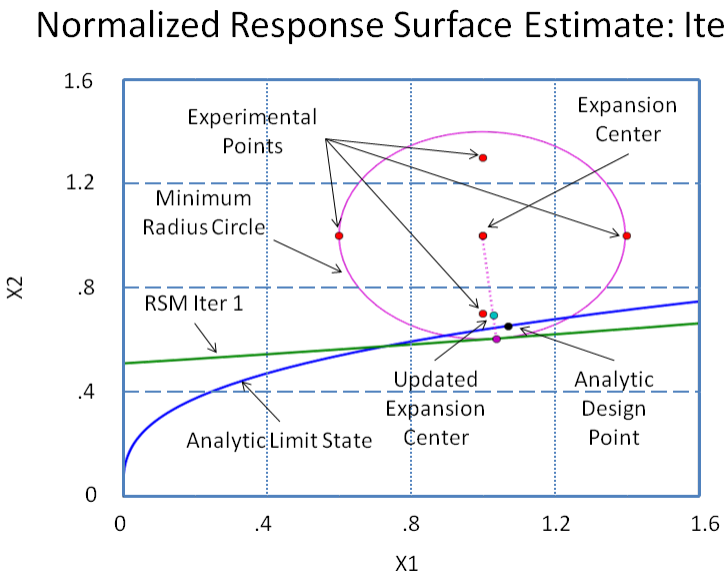
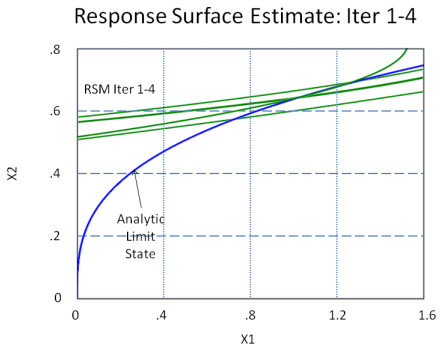


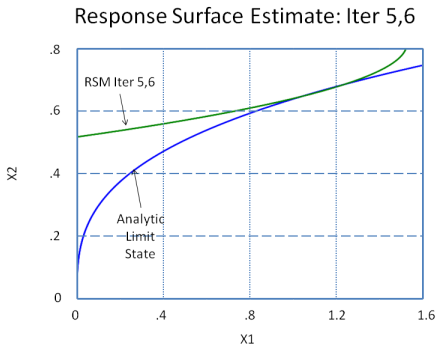
Figure 3.3: Normalized RSM Iteration 1

process repeats until the convergence criteria is satisfied.

Using the iterative method to update the design point and generate a new response surface estimate allows an accurate expansion of the limit state around the design point. Figure 3.4(a) and Figure 3.4(b) show the iterative method convergence. Six iterations of the RSM means the analytic limit state function is evaluated $6 \times (2n + 3) = 42$ times. To assess the accuracy of the RSM estimate, the FORM program was run with both the analytic and estimate of the limit state function. In Table 3.1 the analytic and estimate results are summarized. For the analytic function, the FORM converges in approximately thirteen iterations. The reliability index β converges to 2.33 which corresponds to a probability of failure of .99031%. Comparing to the value



(a) RSM Iteration 1-4



(b) RSM Iteration 5,6

Table 3.1: Analytic vs Estimate in FORM

	g_analytic		g_estimate	
Iteration	U	Beta	U	Beta
1	(0, 0)	-	(0, 0)	-
2	(2.36, -5.3)	5.8037	(-0.016, -0.075)	0.0764
3	(0.33, -5.35)	5.3604	(0.016, 0.072)	0.0741
4	(0.405, -4.92)	4.9353	(-0.014, -0.069)	0.0704
5	(0.405, -4.92)	4.3932	(0.013, 0.0606)	0.0621
6	(0.471, -4.37)	4.3932	(-0.014, -0.070)	0.0716
7	(0.519, -3.71)	3.7441	(0.014, 0.063)	0.0645
8	(0.540, -3.02)	3.0698	(-0.014, -0.070)	0.0714
9	(0.548, -2.50)	2.5583	(0.014, 0.0635)	0.0640
10	(0.572, -2.28)	2.3555	(-0.014, -0.070)	0.0714
11	(0.591, -2.25)	2.3314	(0.014, 0.063)	0.0641
12	(0.593, -2.25)	2.3309	(-0.014, -0.070)	0.0714
13	(0.593, -2.25)	2.3309	(0.014, 0.063)	0.0641

of the reliability index given in Rajashekhar [6] the FORM program is again shown to be working correctly. However, the estimate does not converge, and instead bounces from $(-0.014, -0.070)$ to $(0.014, 0.063)$. Because the estimate is only accurate near the expansion center, a starting point of $(0, 0)$ is using the estimate in a region where the estimate is an inadequate approximation to the actual limit state. Instead, start the MPP search at the RSM final expansion center, converted to U space. The final design point is given by $(1.034, .644)$ in the standard space yielding a value of $(1.034 \times 10^{-03}, 161.21)$ in the original design space. Converting to standard normal distributions gives a starting point, in U space, as $(.1725, -2.368)$. Table 3.2 shows the effect of changing the initial starting point for the MPP search algorithm. With the updated

Table 3.2: MPP with Updated Staring Point

	g_analytic		g_estimate	
Iteration	U	Beta	U	Beta
1	(.173, -2.368)	2.374	(.173, -2.368)	2.374
2	(0.634, -2.288)	2.374	(0.533, -2.313)	2.374
3	(0.585, -2.257)	2.374	(0.551 -2.278)	2.374
4	(0.593, -2.254)	2.332	(0.569, -2.272)	2.343
5	(0.593, -2.254)	2.331	(0.572 -2.272)	2.343
6	(0.593, -2.254)	2.331	(0.572 -2.2715)	2.342
7	(0.593, -2.254)	2.331	(0.5721 -2.2716)	2.342
8			(0.5721 -2.2716)	2.342

starting point, the limit state estimate converges in approximately eight iterations, and the reliability index is calculated as 2.34. The failure probability using the reliability index estimate is given by .99% which is extremely close to the failure probability given by the analytic limit state function.

3.3.3 Comparison with Reference

An issue in reworking the reference example is shown in Figure 3.3.3. The plot is pulled from Rajashekhar [6], but the identified contour appears to correspond to a different limit state function than that listed in the example. Figure 3.3.3 shows the analytic zero contour, design point, and response surface approximations given by Rajashekhar [6]. However, picking points along the

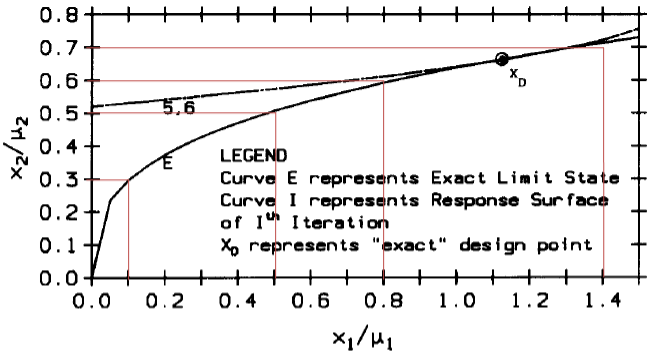


Fig. 6. Convergence—Approach A-0 $-h_i = 1.0$.

X1/μ1	X1	X2/μ2	X2	g_exact
.1	1e-4	.3	75	0.7384
.5	5e-4	.5	125	-0.6793
.8	8e-4	.6	150	0.7384
1.4	1.4e-3	.7	175	-1.070

Figure 3.4: Reference Comparison Discrepancies [6]

$g = 0$ contour, converting them into the standard space, and plugging them into the analytic function of the response surface yields the values shown in the table of Figure 3.3.3. From the table, it is clear that the figure provided in Rajashekhar [6] does not represent the limit state function given. However, the reliability index and failure probability values in the text of Rajashekhar [6]

appear to be correct and were verified by following the methodology discussed.

3.4 Response Surface FORM Comparison

To further validate the RSM, the two examples discussed in Chapter 2 are reworked using the response surface estimate. The results of the FORM program using the response surface estimate will be compared to the analytic results discussed in Chapter 2.

First, consider the cantilever beam discussed in Section 2.4.2. The analytic limit state is repeated here as

$$g = D_0 - \frac{4L^3}{Ewt}\sqrt{(\frac{\mathbf{P_y}}{t^2})^2 + (\frac{\mathbf{P_x}}{w^2})^2} \tag{3.4.1}$$

The random variable definitions in Appendix B.2 code must be changed appropriately, and the analytic limit state programmed into the RSM code, Appendix B.4. Running the code calculates the quadratic response surface estimate parameters and the final expansion center. Plugging in the estimate into the FORM code, Appendix B.5, and running the code yields a β of 1.744 which is equivalent to using the analytic limit state function. The estimate coefficients, final expansion center, β and U-point are summarized in Table 3.3 From Table 3.3 it is clear that using the response surface estimate yields the same FORM solution in the cantilever beam example described previously. Similarly, the solder void growth problem presented in 2.5 is reworked by changing the set up program and the limit state function in the RSM code. Plugging the response surface coefficients and final expansion center, converted

Table 3.3: RSM Cantilever Beam Summary

RS Coefficients	Final Expansion Point	β	U-Space
a=2.6651	(673.38, 1019.46)	1.7444	(1.7367, 0.1640)
b1= -3.3957e-03			
b2=-4.6045e-05			
c1= -3.6855e-07			
c2=-1.5817e-07			

to U-space, into the FORM generates a β of 1.17 which is again equivalent to the analytic formulation. The summary of the response surface coefficients, final expansion point, β, and U-point are summarized in Table 3.4.

Table 3.4: RSM Solder Void Growth Summary

RS Coefficients	Final Expansion Point	β	U-Space
a=4.7086e-01	(300.150, 0.7809)	1.17	(1.127, -0.32)
b1=-8.9955e-03			
b2= 4.06144460			
c1= 5.4210e-20			
c2= -1.5456			

The current section showed that using the RSM to estimate the analytic limit states in Chapter 2 examples produced the equivalent FORM results. The following section discusses methodology considerations that are important for implementation in more complex systems.

3.5 Methodology Comments

Two methodology considerations need to be discussed further. In Section 3.1 of Rajashekhar [6] a comment is made that if the variables are non-Gaussian

then the design variables can first be transformed and then RSM procedures follow as discussed. However, care needs to be taken in the transformation procedure. Because the limit state function is only defined for the design variables, the response surface must first be generated in the original design space and then transformed. In Rajashekhar [6] there is not a detailed discussion of non-Gaussian RSM, and if the underlying variables are highly non-Gaussian a more detailed literature search needs to be completed on the methodological steps.

Work also needs to be done to determine the appropriate order of the response surface estimate. If the order is to high the system will become ill conditioned, and to low creates an inaccurate approximation. Additionally, without testing, the effect of cross terms can not inherently be neglected.

Chapter 4

RS and FORM of a Dynamic Model

4.1 Introduction

In Chapter 2 the general reliability problem was outlined, and the First Order Reliability Method (FORM) was derived and demonstrated. The analysis used a search algorithm for the Most Probable Point (MPP) based on the Taylor series expansion of the limit state. Using the results of the FORM a sensitivity analysis was performed to characterize the importance of variable variation on system reliability. A Python program was developed to perform the FORM given an analytic limit state function, and to calculate the sensitivity measures for the respective variables. The program results were validated by reworking sample problems of Du [3] and Robinson [7].

A Response Surface Method (RSM) was described in Chapter 3 to generate an explicit approximation to an implicit function. The approximation is assumed to have a quadratic structure. Algebraic equations are generated by evaluating the limit state function at a set of experimental points. The resulting equations are solved to yield the approximation coefficients. The iterative method was demonstrated with a problem discussed in Rajashekhar [6]. While there were some discrepancies with the published results, the methodology was

verified. Additionally, the examples considered in Chapter 2 were reworked using the RSM to verify results agreed with the analytic FORM results.

The focus of this chapter is to demonstrate the RSM and FORM techniques on a dynamic system model. A dynamic system presents unique challenges due to the implicit limit state formulation. Unlike previous examples, there is not an underlying explicit limit state formulation for result verification. The model of interest is a pendulum cart assembly. The system was chosen for two specific reasons. First, analytic equations can be derived, Appendix A.5, and second, the two degree of freedom model has three natural limit states of interest. Section 4.3 discusses the quantitative formulation of the limit state functions. In particular, the three limit state functions are given as the maximum pendulum angle, the maximum system velocity, and the combined requirement that neither the maximum angle or velocity is exceeded. To determine the impact of system variable uncertainty on the probability of system failure, a sensitivity analysis, Section 4.4, is performed. By neglecting parameters with less impact on the system reliability, the system order can be reduced to simplify the analysis.

In Section 4.5 the RSM is applied to the reduced system to generate the response surface formulation for the limit states of interest. The FORM is then carried out, Section 4.6, by using the approximations derived in 4.5. A discussion of the results, formulation considerations, and verification issues are addressed in Section 4.7.

4.2 Equations of Motion

The analytic equations of motion for the pendulum cart system, Figure 4.1, are derived from first principles in Appendix A.5. A general multibody

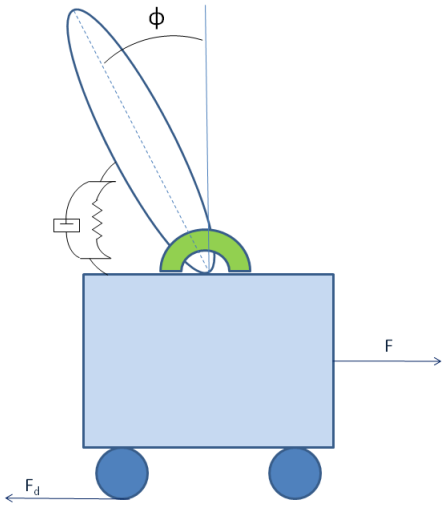


Figure 4.1: Pendulum Cart System

formulation was used to derive the equations of motion for the two body system. The general approach was chosen by the author to gain experience handling general multibody problems. It is worth noting that other approaches may simplify the analysis for the presented problem, for example Bond Graph Methods or Lagrange formulations, and the derivations will yield equivalent results. The state space equations derived in Appendix A.5 are repeated here

for reference:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\omega} \\ \dot{X} \\ \dot{V} \end{bmatrix} = M^{-1} \begin{bmatrix} 0 & 1 & 0 & 0 \\ (m_p g l_1 - k) & -b & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -F_d \end{bmatrix} \begin{bmatrix} \phi \\ \omega \\ X \\ V \end{bmatrix} + M^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} F \qquad (4.2.1)$$

where

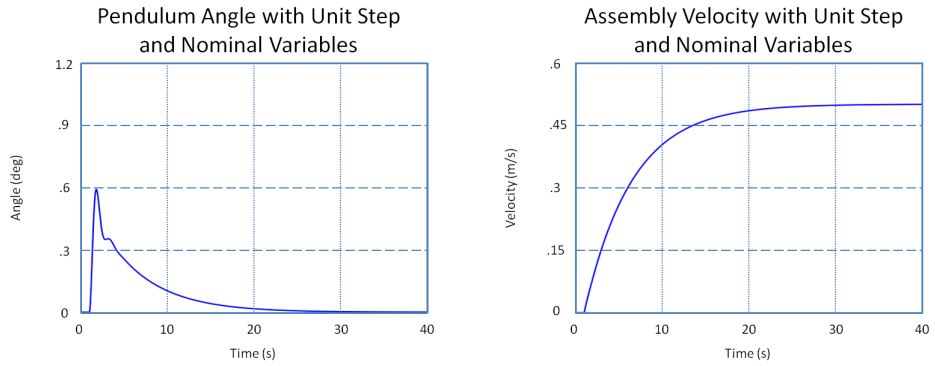
$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & (I_{zz} + m_p l_1^2) & 0 & m_p l_1 \\ 0 & 0 & 1 & 0 \\ 0 & -m_p l_1 & 0 & (m_c + m_p) \end{bmatrix} \qquad (4.2.2)$$

Equations 4.2.1 and 4.2.2 are the linearized equations of motion for the inverted pendulum cart system shown in Figure 4.1 The equations of motion were programmed into a Python script, Appendix B.3, which solves the set of linear differential equations with given simulation parameters. Nominal simulation values are provided in Table 4.1. The system response to a unit step input

Table 4.1: Nominal Ensemble Simulation Values

Parameter	Value	Units
g	9.81	m/s^2
mc	10.0	Kg
mp	1.0	Kg
k	10.0	(N m)/rad
b	1.0	(N m)/(rad/s)
h	.5	m
I_{zz}	.08333	Kg m^2
F_d	1	N/(m/s)
F	1	N

force, F, is shown in Figure 4.2(a) and 4.2(b). Figure 4.2(a) shows that the input force induces a damped oscillatory motion in the pendulum. The motion decays as the pendulum angle approaches a final steady state value. Figure



(a) Pendulum Angle (b) System Velocity

Figure 4.2: Dynamic Response to a Unit Step Force

4.2(b) shows how the system velocity changes due to a unit step force input. The system velocity increases until the damping force equals the applied force, and a steady state velocity is reached. Section 4.3 will discuss the system limit states and derive quantitative formulations.

4.3 Limit State Derivation

In the previous section the dynamic equations were summarized, and general system performance was discussed. The applied force causes the system to accelerate, reaching a steady state velocity when the application force is balanced by the rolling resistance. The system acceleration also induces a dynamic response in the inverted pendulum. The overall system is naturally described by three particular limit state functions.

First, a natural limit state specifies the maximum allowable pendulum angle. Because of the small angle assumptions made in the initial derivation, if

the angle of the pendulum exceeds approximately five degrees, then the small angle approximations break down. The first limit state of interest, Case 1, will consider the maximum pendulum angle. An upper bound of one degree is specified as the limit state for the presented analysis. One degree is used, instead of five, so that the model is valid throughout the region of interest. If the maximum angle was chosen as five degrees, then the experimental points, chosen during the RSM, may lie outside of the model viability region. The nonlinear model would then need be considered since the linear approximation may no longer valid. The limit state function for Case 1 is 4.3.1

$$g_1 = \theta_{lim} - \phi_{sim} \tag{4.3.1}$$

where θ_{lim} is the maximum allowable pendulum angle, one degree, and ϕ_{sim} is the maximum simulated pendulum angle with given parameter values.

The application force, F, causes the system to accelerate until the rolling resistance balances the application force. If there is a maximum system velocity, then variation in the input force or rolling resistance may cause the system velocity to violate the maximum safe speed. Therefore, a second natural limit state function is a maximum system velocity, Case 2. In the following analysis, a maximum velocity of .55 m/s was specified. While the maximum angle had a tractable value assignment, the maximum velocity value was chosen arbitrarily based on convenient parameter values. The limit state function is given by

$$g_2 = V_{lim} - V_{sim} \tag{4.3.2}$$

where V_{lim} is the maximum allowable ensemble velocity, .55 m/s, and V_{sim} is the maximum simulated velocity with given parameter values.

Finally, the third limit state, Case 3, will be the combined requirement that neither the pendulum angle nor the ensemble velocity exceeds their maximum values. The limit state function is then

$$g_3 = min(g_1, g_2) \tag{4.3.3}$$

where g_1 and g_2 are defined by 4.3.1 and 4.3.2. The analysis of Case 3 will illuminate some methodology considerations that will be further discussed in Section 4.7.

The three limit states of interest are dependent on the simulation results with nominal parameter values given in Table 4.1. However, each parameter can be considered to be a random variable with a given probability distribution. Section 4.4 will identify the most influential parameters through a sensitivity analysis, and the results will be used to reduce the problem order and simplify the analysis.

4.4 Sensitivity Analysis

From a design perspective, each variable in Table 4.1 has an associated uncertainty. Because the system is composed of discrete elements, a reasonable assumption of variable independence can be made. Initial analysis will assume each system parameter, excluding gravity, can be described as a Gaussian random variable. The variables are assumed to have a .1 coefficient of variation,

and means given by the nominal values. The parameters are expected to be $\pm$ 10% of the nominal sixty eight percent of the time, $\pm$ 20% of the nominal ninety five percent of the time, and $\pm$ 30% of the nominal ninety nine percent of the time. Intuitively, a .1 coefficient of variation represents a reasonable first pass scenario. Unlike the examples considered in Chapter 2 and 3, the design space consists of eight variables instead of two. Because the largest order problem that can be reasonably illustrated graphically is a two parameter design space, a sensitivity analysis is used to reduce the problem order. By comparing the magnitude of the importance factors, the two parameters with the greatest impact on system reliability can be isolated for further analysis.

Due to the higher order design space, code developed previously to generate the response surface approximation, Section 3.3, has to be altered. In the reference code, Appendix B.4, the minimum norm point along the zero contour, $g_{est} = 0$, is found by pulling values from built-in Python contour functions. However, the Python subroutines do not work with higher order functions. To calculate the minimum norm point in the higher dimensional problem, a series of nested for-loops were set to iterate over the design space. Each point in the design space represents a set of simulation parameters and corresponds to a value of g_{est}. Storing the design space points where $g_{est} \approx 0$ and then determining the minimum norm point yielded the design point. The design point is then used to update the expansion center in the iterative RSM.

While straightforward, the nested loop implementation is extremely inefficient for high order problems. If higher order systems are to be routinely

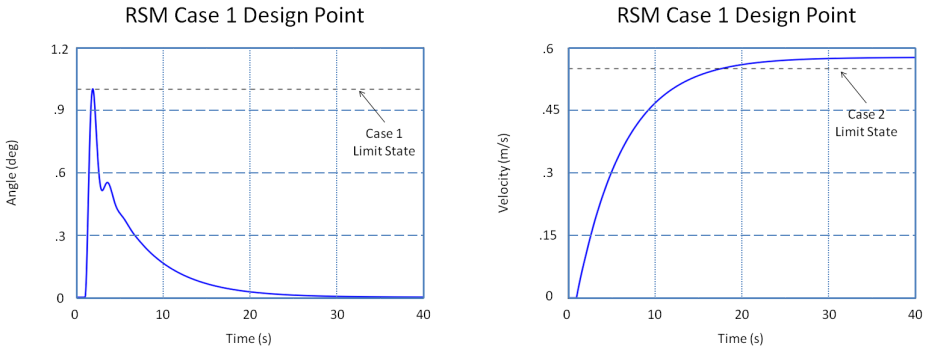
analyzed, the design point calculation needs to be approached with more computationally efficient algorithms. Running the augmented code, the response surface coefficients and design point for Case 1 and 2 are found, Table 4.2. Case 3 will be discussed in the reduced space, Section 4.5, to illustrate methodology considerations when analyzing discontinuous limit state functions. The

Table 4.2: Response Surface Coefficients in Higher Order Space

	RS Coefficient Case 1	RS Coefficient Case 2
a	-7.76787289e+00	-5.68459472e-01
b1	2.13479663e-01	-1.03920236e-03
b2	2.57135300e+00	1.60725124e-04
b3	1.06852377e+00	5.25622789e-07
b4	7.23618317e-01	1.62580962e-06
b5	6.07999008e+00	2.80542924e-06
b6	-6.65287174e-01	1.37973769e-05
b7	5.36026427e-02	8.37981051e-01
b8	-8.20217310e-01	-4.99584361e-01
c1	-6.58007573e-03	6.78259293e-05
c2	-2.23793928e+00	6.68423244e-05
c3	-4.61345589e-02	-2.25052226e-08
c4	-1.78007107e-01	-7.61365831e-07
c5	-9.70732633e+00	-4.86881200e-06
c6	1.64675696e+00	-8.23615231e-05
c7	-1.41514134e-03	-1.38531294e-01
c8	-2.58635968e-07	1.41666724e-07

response surface is assumed to be a quadratic with linear coefficients b_i and squared coefficients c_i. The final design point returned by the RSM can be used as a sanity check to verify the code operation. Since each point in the design space represents a simulation run, a simulation with the design point values should push the simulation output to the limit state value.

Two simulation runs are shown using the design point for Case 1 and Case 2, respectively. Figure 4.3(a) shows the pendulum motion with the Case 1 design point. The simulation output shows that the pendulum angle reaches the maximum angle of one degree as expected. Figure 4.4(b) shows the system velocity with the Case 2 design point. The design parameters are such that the system velocity approaches the maximum allowable system velocity of .55 m/s. From the simulation results shown in Figure 4.3 and 4.4 the design points specified in Table 4.3 lie on the zero contour of the respective limit state function.

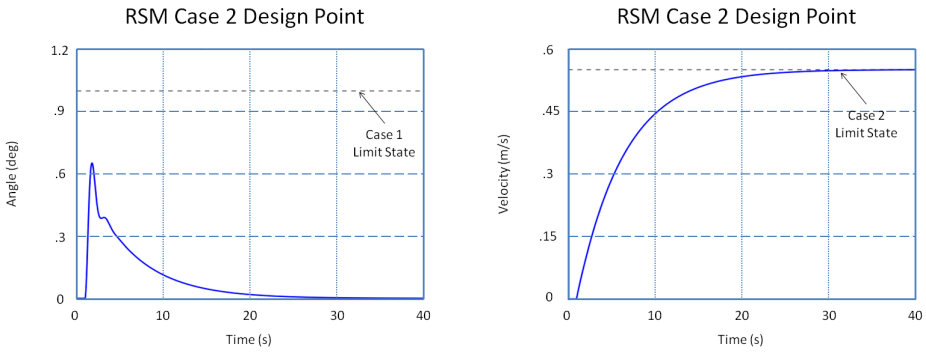


(a) Pendulum Angle

(b) System Velocity

Figure 4.3: Design Point Case 1

Using the FORM code developed previously with the response surface estimate coefficients in Table 4.2, the reliability parameter and importance factors of the design variables can be calculated. The results of the FORM analysis are shown in Table 4.3. The value β can be related to the probability of system failure, pf by the standard cumulative distribution function $\Phi(-\beta)$.



(a) Pendulum Angle (b) System Velocity

Figure 4.4: Design Point Case 2

Table 4.3: Sensitivity Results

Parameter	Case 1		Case 2	
B	1.41	-	.649	-
pf	7.93%	-	25.79%	-
	Sensitivity (γ)	Design Point	Sensitivity (γ)	Design Point
Mc	.214	10	.004	10
mp	.563	1.03	3e-4	1
k	.538	9.30	9e-7	10
b	.093	1.03	1e-7	1
h	.545	.5365	1e-6	.5
I_{zz}	.008	.0813	8e-9	.08501
F_d	.024	1.854	.781	2
F	.204	1.22	.625	1.1

The probability of failure, with given parameter variation, is approximately eight percent in Case 1 and twenty six percent in Case 2.

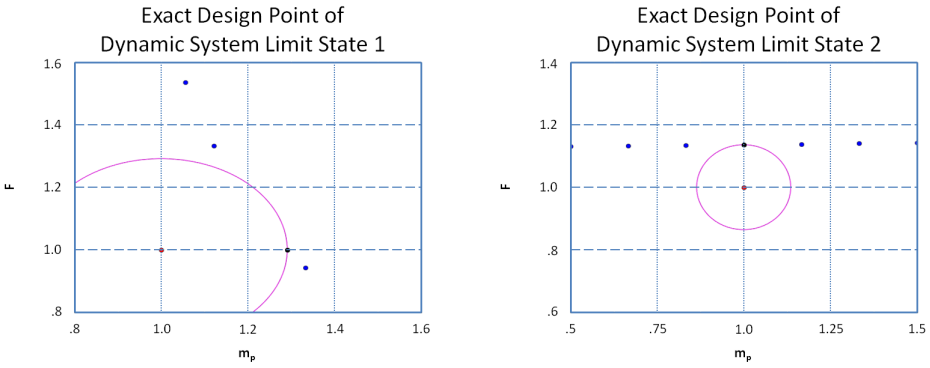
The results of the FORM readily facilitate a sensitivity analysis. The sensitivity, γ, of each parameter is also shown in Table 4.3. In Case 1 multiple system parameters affect the maximum pendulum angle. The greatest affects are due to changes in the pendulum mass, the pendulum center of gravity height, and the spring constant. For Case 2 the only parameters of significance are the applied force, and the damping force. Intuitively, the relative magnitude of sensitivities for the respective values are in line with expected results.

The two parameters picked for system reduction were m_p and F. While m_p has the highest magnitude importance factor for Case 1, the maximum pendulum angle is also greatly affected by variations in the center of gravity height and the spring constant. F was chosen over F_d for the reduced space analysis even though F_d has a larger importance magnitude in Case 2. The choice was made because the damping force variations have negligible impact on Case 1 reliability, whereas the application force, F has a significant impact on maximum pendulum angle. The goal was to chose parameters that had the greatest impact, across the board, on the three limit states of interest. Section 4.5 derives the response surface estimates based on the reduced design space.

4.5 RSM

Using the reduced design space discussed in Section 4.4, the response surface estimates can be derived using the procedural steps laid out in Section 3.2. The two random variables of interest were identified as the applied force, F, and the pendulum mass, m_p. Both random variables have a mean of one, N or kg, and .1 coefficient of variation. Populating the experimental point matrix, 3.2.2, yields the expansion points of interest for the first iteration of the limit state function.

Consider Case 1 and 2, described in Section 4.3, which represent the maximum pendulum angle and maximum ensemble velocity, respectively. Figure 4.5 shows the "exact" design points for Case 1 and 2, Figure 4.5(a) and 4.5(b), respectively.



(a) Case 1 (b) Case 2

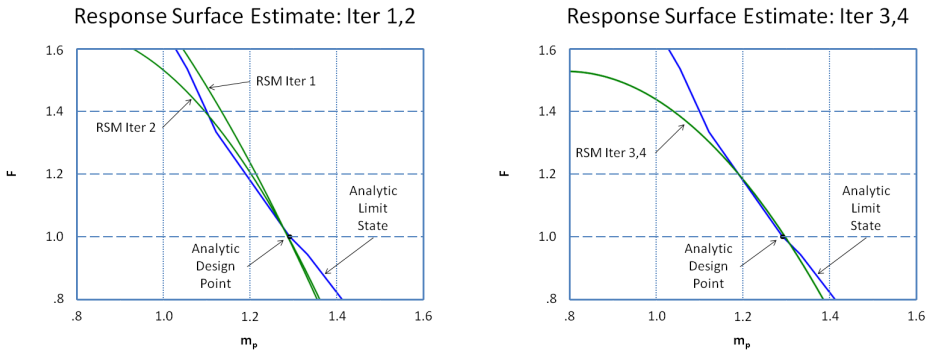
Figure 4.5: Exact Design Point

In the reduced design space, the Python mesh and contour functions can be used to determine the analytic design point by pulling values from

these subroutines. However, even with the simplified design space and simple model under consideration, the design space has to be coarse to allow for acceptable simulation run times. The overall simulation time grows rapidly because each point in the design space requires its own simulation run. For the following plots, a 900 point design space was considered. The contour of the "exact" limit state is shown by blue dots. Notice that the number of points along the $g = 0$ contour are small because the design space is coarse. The norm is calculated at each point along the $g = 0$ contour, and the point with the minimum norm is given as the "exact" design point. The "exact" design point is the intersection of the minimum radius circle, centered at the variable means, which intersect the zero contour. Graphically, it is obvious that the design space coarseness skews the results of the "exact" design point calculation. If the "exact" point needs to be calculated, then the design space density would have to increase, or interpolation between identified limit state points would need to be performed.

The response surface estimate can be generated as a locally accurate approximation to the exact limit state function. Figure 4.6(a) and 4.6(b) demonstrate the iterative procedure to determine the limit state estimate for Case 1. Similarly, the procedure can be repeated for the Case 2 limit state, Figure 4.7. Finally, the response surface expansion for Case 3, the combined limit state requirement can be seen in Figure 4.8. Unlike the RSM procedure shown in Case 1 and 2, the response surface estimate is a poor approximation in the region about the exact design point, shown by the black dot. The

coefficient values and design point estimate can then be plugged directly into the FORM program developed previously. The response surface coefficients



(a) Iteration 1,2 (b) Iteration 3,4

Figure 4.6: Case 1 RSM Iterations

for the three cases of interest are summarized in Table 4.5. The coefficient values will be used in the FORM Python code, Appendix B.5, to determine the system reliability.

Table 4.4: RSM Coefficient Summary

	Coefficient Limit State 1	Coefficient Limit State 2	Coefficient Limit State Combined
a	2.24488602e-01	5.44723209e-01	-2.78243902
b1	3.08789648	4.90572449e-03	6.40602208
b2	-9.46320550e-01	-4.84190622e-01	-4.83463606e-01
c1	-1.95157214	3.57186312e-04	-3.05118722
c2	-3.62863068e-07	2.24854583e-06	2.24854583e-06

Graphically, Figure 4.6 and 4.7 show that the quadratic response surface estimate provides a reasonable approximation in a region about the design point. However, from Figure 4.8, the generated response surface for Case 3

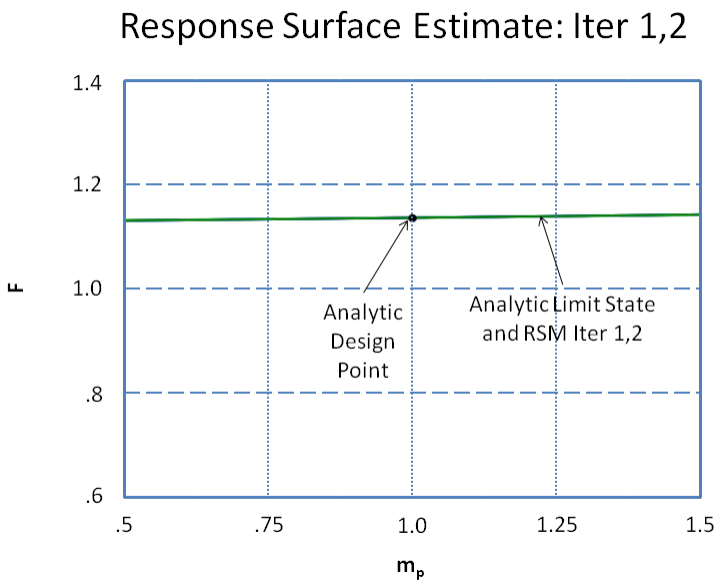
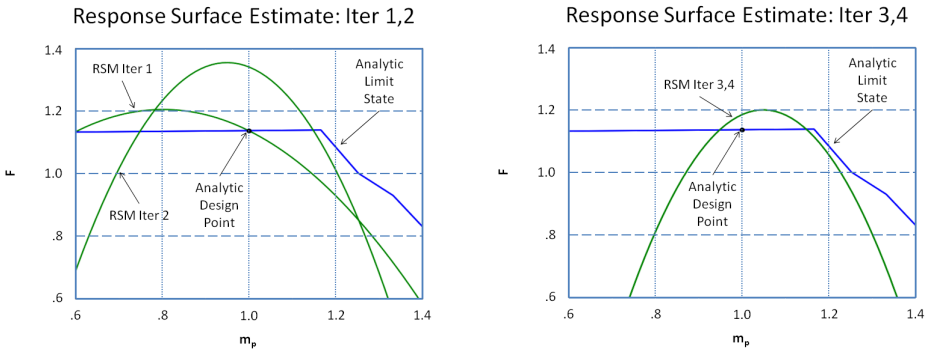


Figure 4.7: Case 2 RSM Iteration 1,2



(a) Iteration 1,2

(b) Iteration 3,4

Figure 4.8: Case 3 RSM Iterations

does not appear to provide a good approximation of the actual response surface. The response surface inaccuracy issues will be discussed further in Section 4.7. Section 4.6 performs the FORM using the analytic response surface estimates summarized in Table 4.5.

4.6 FORM

In Section 4.3 three particular limit states of interest were identified for the cart pendulum ensemble, and Section 4.5 generated response surface estimates from the implicit limit state functions following the RSM. The coefficient values for the three response surface estimates were summarized in Table 4.5. The three analytic response functions were then input into the FORM Python program, Appendix B.5.

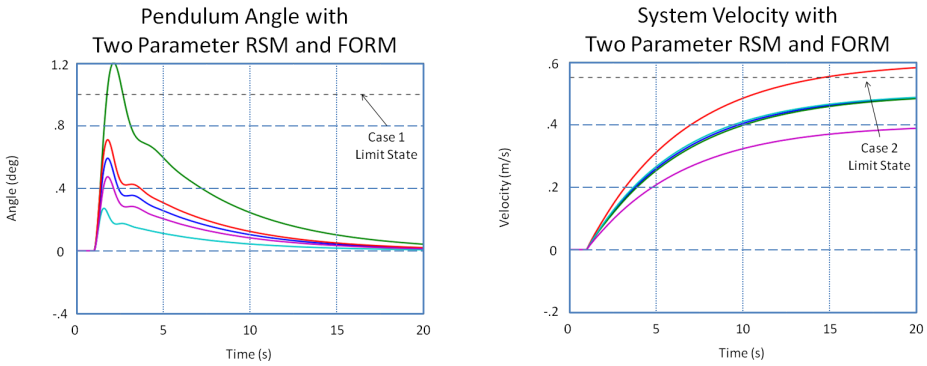
The results of the FORM for the three analytic are summarized in Table 4.6. Notice that the initial point for the MPP search algorithm is given by the design point generated by the RSM. Because the response surface is only valid in a small region about the design point, if the MPP search algorithm starts at the variable mean, $(0, 0)$ in U-space, the FORM program will not converge. Notice that, to an order of magnitude, the results shown in Table 4.6 agree with what would be intuitively expected based on Figure 4.9(a) and 4.9(b). The figures show the simulation runs of the experimental points for the first response surface expansion iteration. Recall that the expansion points, as discussed in Section 3.2, of the first iteration are centered about the variable mean, and the additional points are found by adding and subtracting

Table 4.5: FORM Results Summary

	Case 1		Case 2		Case 3	
Iteration	U	Beta	U	Beta	U	Beta
1	(1.27, 1.06)	1.65	(1.00, 1.14)	1.52	(1.14,1.137)	1.62
2	(1.55, .406)	1.60	(-0.042, 1.76)	1.76	(1.33, .293)	1.36
3	(1.41, .329)	1.44	(-0.031, 1.36)	1.36	(1.12, .205)	1.34
4	(1.40, .346)	1.44	(-0.032, 1.36)	1.36	(1.08, .247)	1.11
5	(1.40,.347)	1.44	(-.031, 1.36)	1.36	(1.08,.256)	1.11
Failure Probability	7.49%		8.69%		13.35%	
Reliability	92.51%		91.31%		86.65%	

two standard deviations from the variable mean. In both figures the limit state function identified in Section 4.3 is shown as a dotted black line. From Figure 4.9(a), Case 1, a two sigma variation can cause the pendulum angle to exceed the defined limit state. A greater than one sigma variation can occur approximately 16% of the time and therefore a failure probaility in the 10% range would be reasonable. A similar order of magnitude analysis can be applied to Case 2. The order of magnitude analysis is the most intuitive analysis that can be used to verify the FORM results.

The previous section carried out the FORM based on the explicit response surfaces derived in Section 4.5. From the results shown in Table 4.6 the failure probability of the system is approximately 10% for the limit cases identified. While the Reliability methods discussed in the current work provide a rapid and flexible analysis tool, there are some issues associated with the methodology application which need to be further discussed.



(a) Pendulum (b) Velocity

Figure 4.9: Response at First Iteration Expansion Points

4.7 Methodology Comments

Using the response surfaces generated in Section 4.5, the FORM was carried out in Section 4.6 and the probability of system failure was calculated for each limit state of interest. Visually, the response surface estimates for the Maximum Pendulum Angle and Maximum System Velocity, Figure 4.6(b) and 4.7 respectively, provide an adequate approximation to the analytic limit state in a region about the analytic design point. However, the combined case, Figure 4.8(b), does not appear to be a good approximation of the analytic function in a region about the analytic design point.

Additionally, recall that the response surface is obtained through multiple evaluations in the design space of interest, i.e. the design space is subdivided and simulations are performed at each point to generate the response surface. Because a simulation must be done at each point in the subdivided design space, in more complicated systems, the analytic response surface would

be unavailable for comparison.

A series but subtle methodology issue is demonstrated in the computation of the combined limit state, Case 3. To verify the failure probability given by the reliability analysis, an Importance sampling technique needs to be used. Du [3], Robinson [7] and Rajashekhar compared the FORM and Importance sampling results for methodology verification. However, the goal of using Reliability analysis in the current example was to avoid using a computationally intensive sampling method. This initial study was unable to find alternative verification procedures for the Reliability methods discussed. While the Reliability method was shown to be equivalent to sampling method for the problems discussed in Chapter 2 and 3, the Reliability method result has no verification metric. Without a verification method the applicability of Reliability methods to general problem applications is significantly reduced.

Chapter 5

Conclusions and Future Work

The goal of the present work was to demonstrate a First Order Reliability Method (FORM) and discuss the strengths and weakness of the proposed methodology when applied to a dynamic system. Unlike importance sampling, Reliability analysis is an analytic technique which finds a particular point in the design space that can accurately be related to the probability of system failure. In Reliability Analysis a limit state function is defined which subdivides the design space into a safe or unsafe region. Through an iterative method, which uses a locally accurate Taylor expansion of the limit state function, the Most Probable Point (MPP) is found. The MPP represents the probability of system failure. Through the references of Du [3] and Robinson [7] a FORM program was tested and verified.

Dynamic systems create a unique challenge for Reliability methods due to the implicit limit state structure. To generate an explicit function from an implicit formulation, a Response Surface Methodology (RSM) was implemented following the work of Rajashekhar [6].

The current work applied the methodology to a dynamic system case study to asses the methodology strengths and weaknesses. The pendulum cart

system had three limit state functions of interest. Visually, the RSM provided a good approximation, in the region of the design point, for two of the three limit states. However, for the third case, the RSM did not visually provide a good approximation to the limit state zero contour. The poor response surface convergence illuminated a major issue with the methodologies previously discussed.

The reference problems provided in Chapter 2 and 3 were studies reported in the literature where a comparison of the FORM result was shown to agree with the Importance sampling result. However, the goal of the current work was to avoid using a computationally intense sampling technique. Currently there do not appear to be alternative methods for FORM verification. As the simulation and failure modes become more complicated, there need to be techniques to determine that the FORM is converging to the same solution that would be generated through a sampling technique. However, this initial study could not identify intermediate verification procedures.

Once a reliable convergence verification metric can be established, Reliability analysis is a flexible technique that, in future work, could be applied to a wide range of applications. Potential areas include optimal control, path planning and software verification. Optimal control focuses on finding a control input that minimizes a performance function. Reliability Analysis could augment the performance function to reflect the user's desire to maintain an acceptable level of system reliability throughout the operation. Additionally, Reliability Analysis could be used to augment path planning algorithms by

providing a failure analysis for given terrain.

Appendices

Appendix A

Dynamic Analysis

A.1 Introduction

The following appendix provides a concise review of basic dynamic principles. The approach is a general analysis methodology chosen by the author for it's flexibility to more advanced problem formulations. It should be noted that there are alternative analysis methods that may simplify analysis based on the particular problem of interest. The appendix follows the work of Ardema [1] and outlines a general problem approach to dynamic analysis. Two specific examples are considered to demonstrate the dynamic analysis methodology. A bicycle model is considered first to verify the methodology with a single body problem before extending the methodology to a multi body pendulum cart system.

A.2 Basic Kinematic Equations

Consider the schematic shown in Figure A.1. The position vector $r_{OA} = r_{OB} + A$ and the rate of change of the position vector r_{OA} is given by

$$\frac{\mathrm{D}}{\mathrm{Dt}} r_{OA} = \frac{\mathrm{D}}{\mathrm{Dt}} r_{OB} + \frac{\mathrm{D}}{\mathrm{Dt}} A \tag{A.2.1}$$

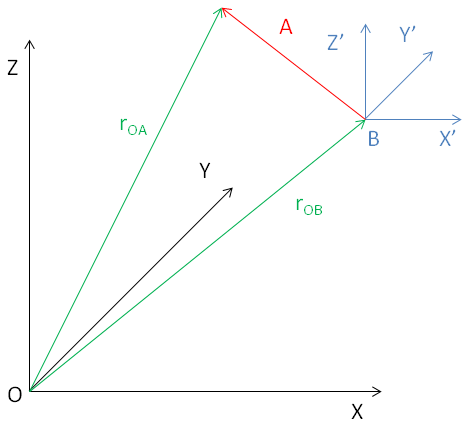


Figure A.1: A vector viewed in a non-inertial frame

where $\frac{\mathrm{D}}{\mathrm{Dt}}$ represents the rate of change of the vector viewed in an inertial frame. However, A is a vector viewed in an arbitrarily rotating translating reference frame, and therefore the rate of change of A, viewed from the B reference frame is given as

$$\frac{\mathrm{D}}{\mathrm{Dt}}A = \frac{\mathrm{d}}{\mathrm{dt}}A + \omega \times A \tag{A.2.2}$$

where $\frac{\mathrm{d}}{\mathrm{dt}}$ is the rate of change of the vector viewed in the non-inertial frame B, and ω is the rotation of the non-inertial frame. Rearranging Equation A.2.1, the rate of change of an arbitrary vector, viewed in an inertial frame, can be equated to the rate of change of the vector viewed in a non-inertial frame.

$$\frac{\mathrm{D}}{\mathrm{Dt}}r_{OA} - \frac{\mathrm{D}}{\mathrm{Dt}}r_{OB} = \frac{\mathrm{d}}{\mathrm{dt}}A + \omega \times A \tag{A.2.3}$$

Realizing that all inertial frames differ by at most a constant velocity, an important simplifying assumption can be imposed. The inertial frame is considered to have the same instantaneous velocity as the reference frame. Assuming

that the inertial frame has the same instantaneous velocity as the reference frame reduces the problem complexity significantly. Considering Figure A.2, the rate of change term $\frac{\mathrm{D}}{\mathrm{Dt}} r_{OB} = 0$ since the reference frame and the inertial frame have the same instantaneous velocity. Equation A.2.3 can be rewritten as

$$\frac{\mathrm{D}}{\mathrm{Dt}} A|_{\text{inertial}} = \frac{\mathrm{d}}{\mathrm{dt}} A|_{\text{reference}} + \omega \times A|_{\text{reference}} \tag{A.2.4}$$

There are two subtleties that need to be addressed here. First, there is

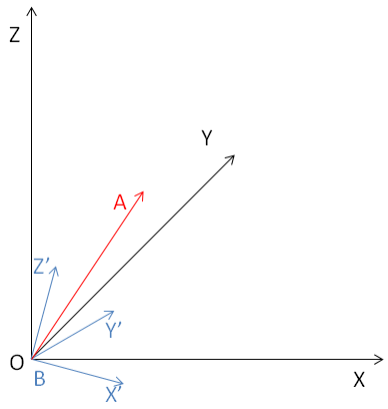


Figure A.2: Non-inertial and Inertial Frame with Equal Instantaneous Velocity

an implicit change of basis vectors in Equation A.2.4. On the left hand side, $\frac{\mathrm{D}}{\mathrm{Dt}} A|_{\text{inertial}}$, vector A is defined in an inertial frame, i, j, k, and on the right hand side, $\frac{\mathrm{d}}{\mathrm{dt}} A|_{\text{reference}} + \omega \times A|_{\text{reference}}$, vector A is viewed from the reference frame, $i'j'k'$. Normally, the change of basis vectors is ignored because the orientation of the selected inertial frame is arbitrary. In other words, the orientation of the inertial frame can be selected such that the axis of the reference frame and the inertial frame are coincident at the instant in time under consideration.

Secondly, a new inertial frame must be considered at every instant in time because the inertial frame and the reference frame will only share the same instantaneous velocity and orientation at the instant shown.

Consider now that both momentum and angular momentum are vectors [1], and that the rate of change of momentum/angular momentum, viewed in an inertial frame, equals the sum of the external forces/torques, respectively. Then A.2.4 can be rewritten as

$$\frac{\mathrm{D}}{\mathrm{Dt}}P|_{\text{inertial}} = \sum F = \frac{\mathrm{d}}{\mathrm{dt}}P|_{\text{reference}} + \omega \times P|_{\text{reference}} \tag{A.2.5}$$

$$\frac{\mathrm{D}}{\mathrm{Dt}}H|_{\text{inertial}} = \sum T = \frac{\mathrm{d}}{\mathrm{dt}}H|_{\text{reference}} + \omega \times H|_{\text{reference}} \tag{A.2.6}$$

where P is the momentum, and H is the angular momentum. Equations A.2.5 and A.2.6 are known as the Basic Kinematic Equations, and will be used extensively Ardema [1].

A.3 Angular Momentum

Due to complexity, a discussion of angular momentum must also be presented. Consider a rotating body with a reference frame such that the rate of change of the position vector in the reference frame to any point on the body is equal to zero. Here after a reference frame with the previous property will be known as a body fixed reference frame. Using a body fixed frame the Basic

[1] angular momentum is actually a pseudo vector

Kinematic Equation, Equation A.2.4, becomes

$$\frac{\mathrm{D}}{\mathrm{Dt}}A|_{\text{inertial}} = \omega \times A|_{\text{reference}} \tag{A.3.1}$$

Angular momentum is defines as the moment of momentum about the origin of the body fixed reference frame, B, and is given by

$$H_B = \sum r_{Bi} \times P_i \tag{A.3.2}$$

where r_{Bi} is the position vector from the reference frame to the ith mass element, and P_i is the momentum of the ith mass element. Note that P_i is defined in the inertial frame, and therefore H_B is the angular momentum in the inertial frame. Additionally, $P_i = m_i \frac{\mathrm{D}}{\mathrm{Dt}} r_{Oi}$ making the angular momentum equation

$$H_B = \sum r_{Bi} \times P_i = \sum r_{Bi} \times m_i \frac{\mathrm{D}}{\mathrm{Dt}} r_{Oi} \tag{A.3.3}$$

recalling that

$$\frac{\mathrm{D}}{\mathrm{Dt}} r_{Oi} = \frac{\mathrm{d}}{\mathrm{dt}} r_{Bi} + \omega \times r_{Bi} = \omega \times r_{Bi} \tag{A.3.4}$$

and using the vector triple product makes the angular momentum equation

$$H_B|_{inertial} = \sum m_i[(r_{Bi} \cdot r_{Bi})\omega - (r_{Bi} \cdot \omega) r_{Bi}] \tag{A.3.5}$$

When Equation A.3.5 is expanded the components of the equations become

$$H_{B_x} = \sum_i m_i(y_i^2 + z_i^2)\omega_x - \sum_i m_i y_i x_i \omega_y - \sum_i m_i z_i x_i \omega_z \tag{A.3.6}$$

$$H_{B_y} = \sum_i m_i(x_i^2 + z_i^2)\omega_y - \sum_i m_i y_i x_i \omega_x - \sum_i m_i y_i z_i \omega_z \tag{A.3.7}$$

and

$$H_{B_z} = \sum_i m_i(x_i^2 + y_i^2)\omega_z - \sum_i m_i z_i x_i \omega_x - \sum_i m_i z_i y_i \omega_y \tag{A.3.8}$$

defining the inertial tensor the angular momentum equation can be presented in a more familiar form as

$$H_B|_{inertial} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{A.3.9}$$

where ω_i is the decomposition of the frame's angular speed vector, viewed in an inertial frame, along the principle axis of the reference frame. Finally, if the reference frame is coincident with the body's principle axis the inertial tensor cross terms vanish and the angular momentum equation becomes

$$H_B|_{inertial} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{A.3.10}$$

Following Ardema [1], the angular momentum of a rigid body with respect to a body fixed frame can be defined in the inertial frame using the familiar inertial tensor.

A.4 Derivation of the Bicycle Model

An intermediate step to demonstrate the application of the dynamic equations is to derive the Bicycle Vehicle model. In the Bicycle Model a vehicle is modeled as a single rigid body moving in a two dimensional plane. The vehicle model has no roll, pitch, or bounce. External forces/torques are not specifically called out since the focus is application of the equations, and a

specific model is not being proposed. Figure A.3 shows a simplified schematic of the vehicle model. Additional assumptions are that the reference frame is a body fixed frame coincident with the principle body axis. Using the previ-

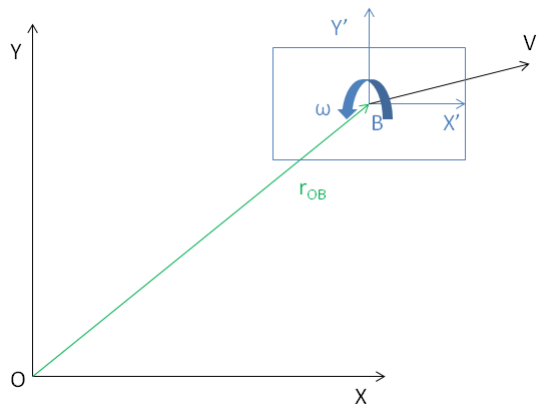


Figure A.3: Bicycle Model Schematic

ous derivations the bicycle model schematic can be simplified. The simplified schematic is shown in Figure A.4. Note that the inertial frame XY and X'Y' are both coincident and share the same instantaneous velocity, at the instant in time shown. Applying equation A.2.5

$$\sum F = \frac{\mathrm{d}}{\mathrm{dt}} P|_{X'Y'} + \omega \times P|_{X'Y'} \tag{A.4.1}$$

where the velocity vector, V, is decomposed along the $X'Y'$ axis, and $P_{X'Y'} = p_x\hat{i'} + p_y\hat{j'}$. Here $\omega = \omega_z\hat{k'}$ due to the initial assumptions. Plugging in and simplifying yields

$$\sum F = \frac{\mathrm{d}}{\mathrm{dt}}(p_x\hat{i'} + p_y\hat{j'}) + \omega_z\hat{k'} \times (p_x\hat{i'} + p_y\hat{j'}) = \dot{p_x}\hat{i'} + \dot{p_y}\hat{j'} + \omega_z p_x\hat{j'} - \omega_z p_y\hat{i'} \tag{A.4.2}$$

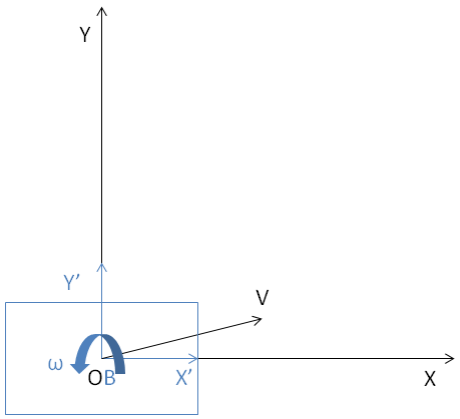


Figure A.4: Equivalent Bicycle Model Schematic

or in matrix form

$$\begin{bmatrix} \sum F_{X'} \\ \sum F_{Y'} \end{bmatrix} = \begin{bmatrix} \dot{p_x} - \omega_z p_y \\ \dot{p_y} + \omega_z p_x \end{bmatrix} \tag{A.4.3}$$

where the forces are decomposed along the reference frame axes. Considering Equation A.2.6, note that the body fixed frame is also coincident with the body principle axis. The angular momentum is then given by

$$H_B|_{inertial} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \omega_z \end{bmatrix} = I_{zz}\omega_z \tag{A.4.4}$$

where $H_B|_{inertial} = H_B|_{X'Y'}$ because the body frame is coincident with the reference frame making A.2.6

$$\sum T = \frac{\mathrm{d}}{\mathrm{dt}} I_{zz}\omega_z + \omega_z \times I_{zz}\omega_z = \dot{\omega_z} I_{zz} \tag{A.4.5}$$

or in matrix form

$$\left[\sum T_{Z'}\right] = \left[\dot{\omega_z} I_{zz}\right] \tag{A.4.6}$$

Equations A.4.3 and A.4.6 represent the governing equations of motion for the rigid body Bicycle Model. The derivation is consistent with accepted equations of motion shown in references.

A.5 Pendulum Cart System

Using the methodology laid out previously in the Appendix, the dynamic equations for the system, Figure A.5, are derived. The force, F, acts through the center of gravity of the cart, and the spring/damper are rotational elements, i.e. the rotational spring generates a force proportional to the angle of rotation, ϕ, while the damper generates a force based on the rate of change of the angle, $\dot{\phi}$. Additionally, F_d, is a damping force which represents a rolling resistance of the system. Two reference frames of interest can be identified, $x'y'$ which stay coincident with the cart pendulum at the connection point between the pendulum and cart, and $x''y''$ which is a pendulum body fixed frame whose origin is coincident with the $x'y'$ reference frame.

The Basic Kinematic Equations (BKE) will be used to equate the sum of the forces/torques acting on an external body with the rate of change of the momentum/angular momentum viewed from the $x'y'$ reference frame. Letting $P_{x'y'}$ represent the momentum and $H_{x'y'}$ the angular momentum of the body viewed in the $x'y'$, the BKEs are given as

$$\sum F = \frac{\mathrm{d}}{\mathrm{dt}} P_{x'y'} + \omega_{x'y'} \times P_{x'y'} \tag{A.5.1}$$

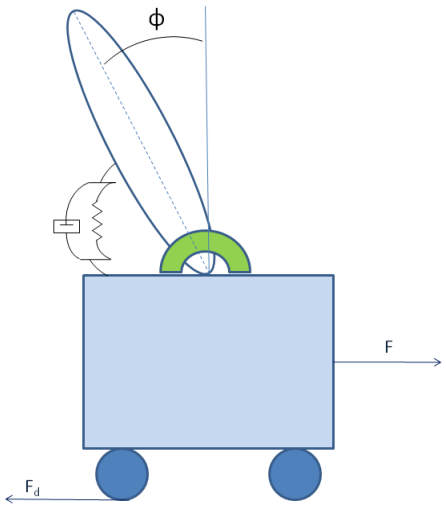


Figure A.5: Cart and Pendulum Ensemble

and

$$\sum T = \frac{\mathrm{d}}{\mathrm{dt}} H_{x'y'} + \omega_{x'y'} \times H_{x'y'} \tag{A.5.2}$$

where $\omega_{x'y'}$ is the rotation of the $x'y'$ reference frame with respect to the inertial frame. In the cart pendulum problem the $x'y'$ reference frame has no rotation with respect to the inertial frame and therefore $\omega_{x'y'} = 0$. Simplifying A.5.1 and A.5.2 yields

$$\sum F = \frac{\mathrm{d}}{\mathrm{dt}} P_{x'y'} \tag{A.5.3}$$

$$\sum T = \frac{\mathrm{d}}{\mathrm{dt}} H_{x'y'} \tag{A.5.4}$$

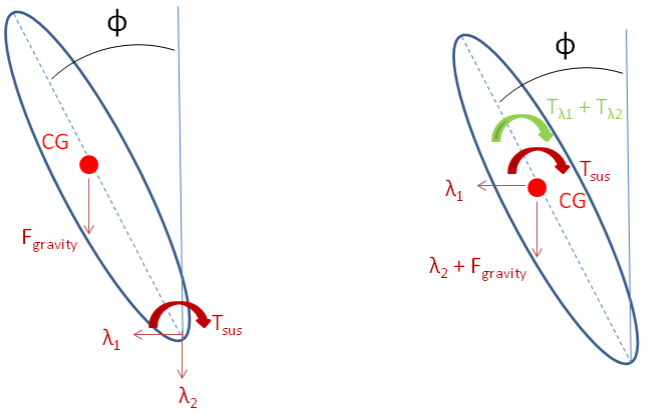
To analyze the multi-body problem, first the equations of motion describing each solid body are derived. The forces and torques acting on each body are then identified and incorporated into the governing equations of mo-

tion for the respective body, Section A.5.1 and A.5.2. In the derivation of the force and torque relations, the effect of each body on the other will be evident. Additionally, two constraint equations, Section A.5.3 will be identified which create algebraic constraints that affect the rigid body motion.

A.5.1 Pendulum

To write the equations of motion of the pendulum, first consider Figure A.6(a) which shows the rigid body pendulum and associated forces. Remembering that a force can be translated to any point on a rigid body with an associated moment couple, the forces acting on the pendulum in Figure A.5.1 can be translated to the pendulum center of gravity as shown in Figure A.6(b). The two contact forces, λ_1 and λ_2, are generated due to the contact point between the pendulum and the cart. Additionally, the contact point is the pivot about which the pendulum rotates. The pendulum weight, F_g, originally acts through the center of gravity, and therefore is unaltered. Translating the two contact forces, λ_1 and λ_2, generates two moment couples, T_{λ_1} and T_{λ_2}. Finally, the suspension, rotational spring/damper, create a torque T_{sus} which is a function of the pendulum angle and rate of change of the angle, ϕ and $\dot{\phi}$ respectively. The sense of the torques acting on the pendulum center of gravity are taken as positive if the torque causes a positive rotation, determined through the right hand rule, about the z' axis,

To use the BKEs discussed in A.5.3 and A.5.4, the momentum, $P_{x'y'}$, and angular momentum, $H_{x'y'}$, of the pendulum viewed in the $x'y'$ reference



(a) Pendulum Force (b) Equivalent Force

Figure A.6: Pendulum Force Diagram

frame must be determined. The momentum of the pendulum is the addition of its forward velocity, P_{bulk}, and the motion of the center of gravity due to the pendulum rotation, P_{roll}. The momentum of the pendulum due to the forward assembly motion is given simply as

$$P_{bulk} = m_p V \hat{i}' \tag{A.5.5}$$

where m_p is the mass of the pendulum and V is the forward velocity of the cart. The momentum of the pendulum due to the change in angle ϕ can be calculated by recalling that the momentum of a body is simply the velocity vector, which is the same for all points on a rigid body, times the mass of the body. To calculate the velocity of the rotating body defined in the $x'y'$ reference frame, a position vector $\hat{r}_{x''y''}$ is defined in the rotating, body fixed, frame $x''y''$. The rate of change of the position vector in the rotating frame is

then equated to the rate of change of the vector in the $x'y'$ reference frame as

$$\frac{\mathrm{d}}{\mathrm{dt}}\hat{r} = \frac{\mathrm{d}}{\mathrm{dt}}\hat{r}_{x''y''} + \omega_{x''y''} \times \hat{r}_{x''y''} \tag{A.5.6}$$

where because the position vector, $\hat{r}_{x''y''}$, is defined in a body fixed frame, by the definition, $\frac{\mathrm{d}}{\mathrm{dt}}\hat{r}_{x''y''} = 0$. The rotation of the $x''y''$ is given by $\omega_{x''y''}$ and is equal to the rate of change of the pendulum angle, ϕ, about the z'' axis. From Figure A.6(b) $\hat{r}_{x''y''} = l_1\hat{j}''$ making Equation A.5.6

$$\frac{\mathrm{d}}{\mathrm{dt}}\hat{r} = \omega_{x''y''} \times \hat{r}_{x''y''} = \dot{\phi}\hat{k}'' \times l_1\hat{j}'' = -\dot{\phi}l_1\hat{i}'' \tag{A.5.7}$$

to equated $\frac{\mathrm{d}}{\mathrm{dt}}\hat{r}$ to $\frac{\mathrm{d}}{\mathrm{dt}}\hat{r}_{x'y'}$ the basis vectors $\hat{i}''\hat{j}''$ need to be changed to $\hat{i}'\hat{j}'$.

$$\hat{i}'' = \cos\phi\hat{i}' + \sin\phi\hat{j}' \tag{A.5.8}$$

plugging in the basis vector change into Equation A.5.7 yields

$$\frac{\mathrm{d}}{\mathrm{dt}}\hat{r}_{x'y'} = -\dot{\phi}l_1\cos\phi\hat{i}' - \dot{\phi}l_1\sin\phi\hat{j}' \tag{A.5.9}$$

and the momentum due to changes the pendulum center of gravity is given by

$$P_{roll} = -m_p l_1\dot{\phi}\cos\phi\hat{i}' - m_p l_1\dot{\phi}\sin\phi\hat{j}' \tag{A.5.10}$$

making a small angle approximation gives

$$P_{roll} \approx -m_p l_1\dot{\phi}\hat{i}' - m_p l_1\dot{\phi}\phi\hat{j}' \approx -m_p l_1\dot{\phi}\hat{i}' \tag{A.5.11}$$

which implies that the total momentum of the pendulum is given by

$$P_{x'y'} = (m_p V - m_p l_1\dot{\phi})\hat{i}' \tag{A.5.12}$$

and the BKE, A.5.3, becomes

$$\sum F = \frac{\mathrm{d}}{\mathrm{dt}} P_{x'y'} = (m_p \dot{V} - m_p l_1 \ddot{\phi})\hat{i}' \tag{A.5.13}$$

The above methodology allows the equations of motion governing the the movement of the pendulum to be related to the sum of the external forces acting on the pendulum. Additionally, the angular momentum of the pendulum yields another set of dynamic equations governing the pendulum motion. To solve for the angular momentum, viewed in the inertial frame, start with the inertial tensor and then consider that the location of the inertial origin can be taken as coincident with the $x'y'$ origin, at the instant shown, and therefore the angular momentum, viewed in the inertial frame, equals the angular momentum viewed in the $x'y'$ reference frame. Considering the inertial tensor yields

$$H_{x'y'} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{A.5.14}$$

where ω_i is the rotation of the body fixed reference frame $x''y''$ viewed from the inertial frame. Because the body fixed frame rotates only about the z'' axis $\omega_x = \omega_y = 0$ and $\omega_z = \dot{\phi}$. Plugging ω into Equation A.5.14 yields [2]

$$H_{x'y'} = I_{zz}\omega_z = I_{zz}\dot{\phi} \tag{A.5.15}$$

and the BKE, Equation A.5.4 becomes

$$\sum T = \frac{\mathrm{d}}{\mathrm{dt}} H_{x'y'} = I_{zz}\ddot{\phi}\hat{k}' \tag{A.5.16}$$

[2]Here $-I_{xy}\ddot{\phi} = I_{xz}\ddot{\phi} \equiv 0$ and $\ddot{\phi} = 0$ is always a trivial solution to any differential equation and has been neglected in further analysis

To determine the sum of the forces and sum of the torques, refer back to Figure A.6(b). The sum of the forces can then be derived via inspection as

$$\sum F_x = -\lambda_1 \tag{A.5.17}$$

$$\sum F_y = -Fg - \lambda_2 \tag{A.5.18}$$

$$\sum T = -(T_{\lambda_1} + T_{\lambda_2} + T_{sus}) \tag{A.5.19}$$

relating A.5.17,A.5.18, and A.5.19 to the respective components in the BKE equations, A.5.13 and A.5.13, yields two dynamic equations

$$m_p\dot{V} - m_p l_1 \ddot{\phi} = -\lambda_1 \tag{A.5.20}$$

$$I_{zz}\ddot{\phi} = -(T_{\lambda_1} + T_{\lambda_2} + T_{sus}) \tag{A.5.21}$$

and one algebraic equation

$$0 = -Fg - \lambda_2 \tag{A.5.22}$$

if a small angle approximation is also used to derive values for the torque terms as

$$T_{\lambda_1} = \lambda_1 l_1 \tag{A.5.23}$$

$$T_{\lambda_2} = \lambda_2 l_1 \phi = -m_p g l_1 \phi \tag{A.5.24}$$

$$T_{sus} = k\phi + bp\dot{h}i \tag{A.5.25}$$

the two governing equations can then be written in a complete form as

$$m_p\dot{V} - m_p l_1 \ddot{\phi} = -\lambda_1 \tag{A.5.26}$$

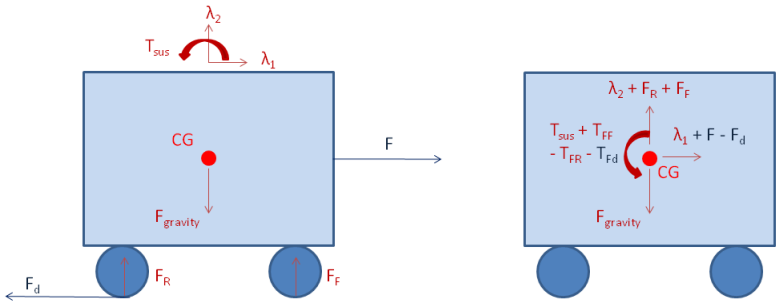
$$I_{zz}\ddot{\phi} = -(\lambda_1 l_1 + -m_p g l_1 \phi + k\phi + b\dot{\phi}) \tag{A.5.27}$$

After deriving the equations of motion for the pendulum, the dynamic equations for the cart need to be derived. The derivation is discussed in Section A.5.2.

A.5.2 Cart

To write the equations of motion for the cart, first consider Figure A.7(a) which shows the rigid body cart and associated forces. Remembering that a force can be translated to any point on a rigid body with an associated moment couple, the forces acting on the cart in Figure A.7(a) can be translated to the cart center of gravity as shown in Figure A.7(b). The two contact forces, λ_1 and λ_2, are again generated due to the contact point between the pendulum and the cart. The cart weight, F_g, originally acts through the center of gravity, and therefore is unaltered. Translating the two contact forces, λ_1 and λ_2, generates one moment couple, T_{λ_1}. Finally, the suspension, rotational spring/damper, create a torque T_{sus} which is a function of the pendulum angle and rate of change of the angle, ϕ and $\dot{\phi}$ respectively. The sense of the torques acting on the cart center of gravity are taken as positive if the torque causes a positive rotation, determined through the right hand rule, about the z' axis,

Using the same procedure outlined in Section A.5.1 the momentum/ angular momentum of the system, viewed in the $x'y'$ frame is determined and related to the sum of the forces/torques acting on the cart, respectively. In the case of the cart, the angular momentum is equivalently defined equal to zero and therefore $H_{x'y'} \equiv 0$. The momentum of the cart is only due to its



(a) Cart Force (b) Equivalent Force

Figure A.7: Cart Force Diagram

forward velocity, P_{bulk} and is given simply as

$$P_{bulk} = P_{x'y'} = m_c V \hat{i}' \tag{A.5.28}$$

where m_c is the mass of the cart and V is the forward velocity of the cart. Relating the BKE A.5.3 and the sum of the forces seen in Figure A.7(b) yields the following dynamic equation for the cart

$$\sum F = P_{x'y'} = F + \lambda_1 = m_c \dot{V} \tag{A.5.29}$$

Following the same procedures outlined in Section A.5.1 the dynamic equations of motion for the cart were derived. In Section A.5.3 the final dynamic equations are solved and put into state space form.

A.5.3 Constraint Equations and Final Equations of Motion

In the Equations A.5.26, A.5.27, and A.5.29 the dynamic equations contained two constraint forces, generated through the contact point between the

cart and the pendulum. Solving for λ_1 in Equation A.5.26 and substituting in to Equations A.5.27 and A.5.29 yields the following two dynamic equations.

$$(I_{zz} + m_p l_1^2)\ddot{\phi} - m_p l_1 \dot{V} = (m_p g l_1 - k)\phi - b\dot{\phi} \tag{A.5.30}$$

$$(m_c + m_p)\dot{V} - m_p l_1 \ddot{\phi} = F \tag{A.5.31}$$

because there are two degrees of freedom, ϕ and V, the implication is that there are four first order differential equations describing the assembly. However, the dynamic equations are a function of $\ddot{\phi}$, $\dot{\phi}$, ϕ, $\dot{V}$, and V the state vector reduces to $\mathbf{x} = [\phi, \omega, V]^T$, and the state space equations can be written as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\omega} \\ \dot{V} \end{bmatrix} = M^{-1} \begin{bmatrix} 0 & 1 & 0 \\ (m_p g l_1 - k) & -b & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \omega \\ V \end{bmatrix} + M^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} F \tag{A.5.32}$$

where

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & (I_{zz} + m_p l_1^2) & m_p l_1 \\ 0 & -m_p l_1 & (m_c + m_p) \end{bmatrix} \tag{A.5.33}$$

Equation A.5.32 represents the equations of motion for the pendulum cart system derived through a general multi body analysis technique.

Appendix B

Python Code

B.1 Introduction

The following appendix contains a sample Python program based on the reduced system pendulum analysis. Each code segment begins with a detailed description of the block function, and supporting comments are provided throughout the block. The code represents a working program to analyze the reduced space pendulum cart system discussed in Chapter 4. The code is broken down into four discrete blocks. The set up code provides the random variable definitions and specifications, the simulation code provides a working simulation for the pendulum cart system of interest, the RSM code generates the response surface approximation for the specified limit states, and the FORM code performs the iterative FORM procedure and sensitivity analysis. While not a comprehensive program, the following code represents the basis for other programs written for this thesis work.

B.2 Set Up Code

```
# Author: Brad Munoz
# Date Last Modified: 10/29/2012

# Function:
# The following program is used to set up
# the parameter values for the RSM and FORM
# programs.

# Revisions:

#Program Start
from numpy import *
def Set_Up():

    #Variable Specification
    num=2 #Number of random variables
    mu_0=1.0; mu_1=1.0 #Variable mean
    V0=.2; V1=.1; #Coeff of variation
    sigma_0=mu_0*V0; sigma_1=mu_1*V1; #Std Dev

    X=zeros(num) #Vector of random variables
    std_dev=zeros([1,num]) #Standard deviation
    mean=zeros([1,num]) #Mean

    #Populate Standard Deviation and Mean matrices
    for i in range(0,num):
        std_dev[0,i]= eval("sigma_"+str(i))
        mean[0,i]=eval("mu_"+str(i))
    return [mean, std_dev]
```

Set_Up_Sim.py

B.3 Simulation Code

```
# Author: Brad Munoz
# Date Last Modified: 10/29/2012

# Function:
# This program will simulate the cart pendulum and return
# the angle and velocity of the ensemble. The equations
# of motion were derived and compared to references and
# then programmed appropriately.

# Revisions:

# Program Start
from numpy import *
from numpy import linalg as LA
from scipy.integrate import odeint

# solve the system dy/dt = f(y, t)
def f(y, t, A_matrix, E_inv,force):
 # Define Current State and Control Input Vectors
 State=array([[y[0]],[y[1]],[y[2]],[y[3]]])
 F=array([[0],[0],[0],[Force(t,force)]])

 # Cacluate Derivative of State Vector
 y_k=dot(A_matrix,State)+dot(E_inv,F)

 # Format Return Array
 return [y_k[0,0],y_k[1,0],y_k[2,0],y_k[3,0]]

def Force(t,force):
 # Define the forcing function
 if t<1.0:
  F=0.0
 else:
  #A=10.0  #Amplitude
  #f=.5 #Hz
  #F=A*sin(2*pi*f*t)
  F=1.*force
 return F

def Simulation(g=9.81,mc=10.0, mp=1.0, k=10.0, b=1.0,
               h=.5, Izz=.0833, F_d=2.0, force=1.):
##########################################################
#Constants Pulled From Similar Academic Example#
```

```
#####################################################
#g=9.81  #gravitational constant (m/s^2)
#mc=10.0 #mass of cart (kg)
#mp=1.0  #mass of pendulum (kg)
#k=10.0  #rotational spring stiffness (N/rad)
#b=1.0   #rotational damping coefficient (N/(rad/s))
#h=.5    #pendulum cg height above cart
#Izz=.08333 #moment of inertia (kg m^2)
#F_d=1.0 #Wheel resistance damping coefficient

# THESE ARE THE DEFAULT SIMULTION VALUES

#####################################################
#Governing Equations
#(Izz+mp*h**2)ddPhi-mp*h*dV=mp*g*h*Phi-Tsus
#-mp*h*ddPhi+(mc+mp)*dV=F-F_d*V

#Tsus=k*Phi+b*dPhi

#(Izz+mp*h**2)ddPhi-mp*h*dV=(mp*g*h-k)*Phi-b*dPhi
#-mp*h*ddPhi+(mc+mp)*dV=F

#States=[X, V, Phi, Omega]: Omega=dPhi, V=dX

#State Space Equations
#|1        0           0      0      | |dPhi   |
#|0   (Izz+mp*h**2)  0   -mp*h   | |dOmega| =
#|0        0           1      0      | |dX     |
#|0    -mp*h          0   (mc+mp)| |dV     |
#
#|0             1   0       0| |Phi   |   |0|
#|(mp*g*h-k) -b   0       0| |Omega| +|0| F
#|0             0   0       1| |X     |   |0|
#|0             0   0  -F_d| |V     |   |1|

#####################################################
 E_matrix=array([[1,        0,           0,          0],
                 [0,(Izz+mp*h**2),    0,      -mp*h],
                 [0,        0,           1,          0],
                 [0,      -mp*h,         0,   (mc+mp)]])

 D_matrix=array([[0,             1, 0,      0],
                 [(mp*g*h-k), -b, 0,      0],
                 [0,             0, 0,      1],
                 [0,             0, 0, -F_d]])
```

```
89
    A_matrix=dot(LA.inv(E_matrix),D_matrix)
91  E_inv=LA.inv(E_matrix)

93  # Initial Conditions
    Phi_o=0.0
95  Omega_o =0.0
    X_o=0.0
97  V_o=0.0

99  # Sprung Mass Simulation
    y0 = [Phi_o, Omega_o, X_o, V_o]# initial condition vector
101 t  = linspace(0.0, 20.0, 1000) # time grid

103 # solve the DEs
    soln = odeint(f, y0, t,args= (A_matrix,E_inv,force))
105 Phi= soln[:, 0]
    Omega = soln[:, 1]
107 X=soln[:,2]
    V= soln[:, 3]
109
    return [t,Phi,V]
```

Pendulum_Cart_Model.py

B.4 RSM Code

```
# Author: Brad Munoz
# Date Last Modified: 10/29/2012

# Function:
# This program generates a Response Surface Approximation
# of an implicit limit state function. The Set_Up_Sim.py
# program is used to specify the number of random variables
# and associated valus, and is shared between mutliple
# programs. The Pendulum_Cart_Model.py program is called
# and the limit state functions are evluated at various
# experimental points. A response surface estimate is
# generated in an iterative process until a convergence
# criteria is satisfied.

# Revisions:

# Program Start
from numpy import *
from numpy import linalg as LA
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import matplotlib.pyplot as plt
from Pendulum_Cart_Model import *
from Set_Up_Sim import *

def Lim_State(X_vector):
    # Simulation run at Experimental Point
    mp=X_vector[0]; force=X_vector[1]
    [t,Phi,V]=Simulation(mp=mp, force=force)

    # Define the three limit states of interst
    g1=1.-max(Phi*180/pi)    #Case 1
##    g2=.55-max(V)            #Case 2
##    g3= min(g1,g2)           #Case 3
    g=g1                     # Select the limit state
    return g

def Exp_Points(mean,std_dev,h):
    # Generate the experimental points at which the
    # limit state will be evaluated
    num=size(mean[0])
    Exp=zeros([2*num+1,num])
```

```
    Exp[0]=mean[0]

    i=0; k=0
    a=h*std_dev[0]
    while k<(2*num):
        for j in range(0,num):
            dExp=zeros(num)
            dExp[j]=a[j]
            Exp[i*num+j+1]=mean[0]+dExp
        a=-a
        k=num+k
        i=i+1
    return Exp

def coefficient(X_Vector,g):
    # Solve the algebraic equations to determine the
    # response surface coefficients (assumes a quadratic
    # structure of the estimate)
    num=size(X_Vector[1])
    colm=num*2+1
    A=zeros([2*num+1,colm])

    for i in range(0,2*num+1):
        A[i,0]=1.0
        k=0

        for j in range(0,colm-1):
            if k<num:
                A[i,j+1]= X_Vector[i,k]
                k=k+1
            else:
                A[i,j+1]= pow(X_Vector[i,j-k],2)
    coeff=dot(LA.inv(A),g)
    return coeff

def Response_Surface_Gen():
    # Generates the Response Surface Estimate
    [Mean,Std_dev]=Set_Up()      # Variables of interest
    X_k=Mean                     # Starting design point

    h_mat=[2.0,1.0]              # Arbitrary multiplicative factor
    Design_norm=1                # Convergence Criteria
    k=0                          # Iteration Counter

#####################################################################
```

```
# ALL OF THIS IS USED TO CALCULATE THE ANALYTIC DESIGN POINT
# FOR THE DYNAMIC SYSTEM RESPONSE SURFACE g=0 WITH GIVEN
# PARAMETER VALUES. WHEN ACTUALLY USING THE PROGRAM THESE
# GRAPHS ARE UNECESSARY AND SHOULD ONLY BE USED FOR
# VERIFCATION IN A SUFFICIENTLY SIMPLE SYSTEMS. IN ACTUAL
# USE THE CODE BELOW WOULDN'T BE USED SINCE THE RSM CALCULATES
# AN ESTIMATE TO THE ANALYTIC SURFACE AND THE CODE BELOW MAY
# BE COMPUTATIONALLY UNFEASIBLE
#########################################################################

# Below is ALL SPECIFIC to the limit state function under
# consideration i.e. the number of design variables, the limit
# state calls, contour generation, etc. in future work this
# section needs to be changed to accomidate changes in the
# limit state definitions, parameters, etc.

# Note all the analysis HAS to be done in a normalized space
# otherwise the percentage change of the magnitude of one
# variable can overshadow a larger percentage change of
# another variable's magnitude
##
##    # Partition the design space
##    xlist=linspace(.5,2.,10)
##    ylist=linspace(0.,3.,10)
##    g=zeros([size(ylist),size(xlist)])
##    x,y = meshgrid(xlist,ylist)
##    test1=shape(x);test2=shape(y)
##
##    # Evaluate the limit state function in
##    # the partitioned design space
##    for i1 in range(0,test1[1]):
##        for j1 in range(0,test2[0]):
##            g[j1,i1]=Lim_State([xlist[i1],ylist[j1]])
##
##    plt.figure(1)
##    CS1=plt.contour(x,y,g,[0])  # Plot the zero contour of
##                                # the limit state
##    #plt.clabel(CS1,inline=1,fontsize=10)#Label plotted contours
##
##    # Extract data from contour plot
##    p_e = CS1.collections[0].get_paths()[0]
##    v_e = p_e.vertices
##    x_e = v_e[:,0]
##    y_e = v_e[:,1]
##
```

```
134 ##     # Normalize standard variables (currently un-used)
    ##     nx_e=x_e; ny_e=y_e                # Normalized contour points
136 ##     nMean=array([[Mean[0,0],Mean[0,1]]])# Normalized Mean
    ##     nexact_design=[nx_e[0],ny_e[0]]      # Starting contour point
138 ##     nexact=sqrt((nx_e[0]-nMean[0,0])**2+
    ##                 (ny_e[0]-nMean[0,1])**2)# Starting norm value
140 ##
    ##     # Calculate the analytic design point by iterating over the
142 ##     # zero contour points derived from the design space
    ##     # subdivision and storing the minimum norm point
144 ##
    ##     for m in range(0,len(x_e)):
146 ##         # Norm of iterated point
    ##         ntestnorm=sqrt((nx_e[m]-nMean[0,0])**2+
148 ##                        (ny_e[m]-nMean[0,1])**2)
    ##         if ntestnorm<nexact:
150 ##             # Store Updated Design Point
    ##             nexact_design[0]=nx_e[m]
152 ##             nexact_design[1]=ny_e[m]
    ##
154 ##             # Update minimum norm
    ##             nexact=ntestnorm
156 ##
    ##     #### Plotting ####
158 ##     # Axis values
    ##     xmin=.5; ymin=.6
160 ##     xmax=1.5; ymax=1.4
    ##     xspace=linspace(xmin,xmax,5)
162 ##     yspace=linspace(ymin,ymax,5)
    ##
164 ##     # Axis Formatting
    ##     plt.figure(2)
166 ##     fig=plt.gcf()
    ##     ax = plt.gca()
168 ##     ax.set_xlim((xmin,xmax))
    ##     ax.set_xticks(xspace)
170 ##     ax.set_ylim((ymin,ymax))
    ##     ax.set_yticks(yspace)
172 ##
    ##     # Plot and Figure Formatting
174 ##     plt.plot(nx_e,ny_e,'bo',linewidth=2.0)
    ##     plt.plot(nexact_design[0],nexact_design[1],
176 ##              'ko', nMean[0,0], nMean[0,1], 'ro',
    ##              linewidth=2.0)
178 ##     circle1=plt.Circle((nMean[0,0],nMean[0,1]),
```

```
##                          nexact,color='m',fill=False)
##     fig.gca().add_artist(circle1)
##     ax.get_xaxis().set_visible(False)
##     ax.get_yaxis().set_visible(False)
##     print xspace, yspace
##
##     #END ANALYTIC DETERMINATION
#####################################################################

    while Design_norm>1e-10:

        # Assign Arbitrary Multiplicatie Factor
        if k<2:
            h=h_mat[0]
        else:
            h=h_mat[1]

        Exp=Exp_Points(X_k,Std_dev,h) #Experimental Points
        shap_Lim=shape(Exp); g=zeros([shap_Lim[0],1])
        x_test=zeros([shap_Lim[0],1])
        y_test=zeros([shap_Lim[0],1])

        # Populate limit state vector
        for i in range(0,len(g)):
            X_Vector=Exp[i]; g[i]=Lim_State(X_Vector)
            x_test[i]=X_Vector[0]; y_test[i]=X_Vector[1]

        # Generate the coefficient values of the response
        # surface estimate
        coeff=coefficient(Exp,g)

        # Determine the minimum norm point by iterating over
        # the response surface estimate zero contour in the
        # design space of interest. Note there are different
        # ways to solve for the response surface estimate min
        # norm point, but the method below is simple and
        # straightforward

        # Design space
        xlist=linspace(.5,3.,200)
        ylist=linspace(0,3.,200)
        x,y = meshgrid(xlist,ylist)

        # Response surface estimate
        g_est= (coeff[0]+coeff[1]*x+coeff[2]*y+
```

```
                coeff[3]*x**2+coeff[4]*y**2)

        # Extract zero contour of limit state estimate
        plt.figure(1)
        CS2=plt.contour(x,y,g_est,[0.])
        p_est= CS2.collections[0].get_paths()[0]
        v_est= p_est.vertices
        x_est = v_est[:,0]
        y_est = v_est[:,1]

        # Normalize variables (currently un-used)
        nx_est=x_est;ny_est=y_est #Normalized contour points
        nX_k=array([[X_k[0,0],X_k[0,1]]]) #Normalized design
        intmed=[x_est[0],y_est[0]]#Starting contour point
        nnorm=sqrt((nx_est[0]-nX_k[0,0])**2+
                   (ny_est[0]-nX_k[0,1])**2) #Starting norm value


        # Calculate the point which minimizes the distance from
        # the variable mean to the zero contour of the limit
        # state estimate
        for m in range(0,len(x_est)):
            # Norm of iterated point
            norm=sqrt((nx_est[m]-nX_k[0,0])**2+
                      (ny_est[m]-nX_k[0,1])**2)
            if norm<nnorm:
                # Store Updated Design Point
                intmed[0]=nx_est[m]
                intmed[1]=ny_est[m]

                #Update minimum norm
                nnorm=norm


        # Evaluate actual limit state
        g_mu=Lim_State(X_k[0])
        g_x=Lim_State(intmed)

        # Update design center through linear interpolation
        X_d=array([[(intmed[0]-X_k[0,0])*
                    g_mu/(g_mu-g_x)+X_k[0,0],
                    (intmed[1] -X_k[0,1])*
                    g_mu/(g_mu-g_x)+X_k[0,1]]])

##        ##########################
```

```
##          #Plotting#
##          ##########################

##          # Axis values
##          xmin=.6; ymin=.6
##          xmax=1.4; ymax=1.4
##          xspace=linspace(xmin,xmax,5)
##          yspace=linspace(ymin,ymax,5)
##
######################################################################
# BELOW PLOTS THE FIRST ITERATION OF THE RSM. THE
# EXPERIMENTAL POINTS ARE SHOWN IN RED, THE EXACT LIMIT
# STATE IN BLUE, THE ESTIMATE IN GREEN, AND THE UPDATE POINT
# IN CYAN. YOU HAVE TO UNCOMMENT THE EXACT DESIGN POINT
# CALULATION TO HAVE THIS RUN.
######################################################################
##          # Axis Formatting
##          plt.figure(2)
##          fig=plt.gcf()
##          ax = plt.gca()
##          ax.set_xlim((xmin,xmax))
##          ax.set_xticks(xspace)
##          ax.set_ylim((ymin,ymax))
##          ax.set_yticks(yspace)
##
##          # Plot and Figure Formatting
##          plt.plot(nx_e,ny_e,'b--',nx_est,
##                   ny_est,'g',linewidth=2.0)
##          plt.plot([intmed[0],nX_k[0,0]],[intmed[1],nX_k[0,1]],
##                   'mo:',x_test,y_test,'ro',X_d[0,0],X_d[0,1],
##                   'co',nexact_design[0],nexact_design[1],
##                   'ko',linewidth=2.0)
##          plt.plot([xmin,xmax,xmax,xmin,xmin],
##                   [ymin,ymin,ymax,ymax,ymin])
##          circle1=plt.Circle((nX_k[0,0],nX_k[0,1]),
##                             nnorm,color='m',fill=False)
##          fig.gca().add_artist(circle1)
##          ax.get_xaxis().set_visible(False)
##          ax.get_yaxis().set_visible(False)
##          print xspace, yspace
##
######################################################################
# THIS SPECIFICALLY PLOTS THE ITERATIONS OF THE RSM.
# YOU HAVE TO UNCOMMENT THE EXACT DESIGN POINT CALULATION
# TO HAVE THIS RUN. THE EXACT LIMIT STATE FUNCTION IS SHOWN
```

```
# IN BLUE AND THE ESTIMATE IS SHOWN IN GREEN.
#########################################################################
##
##          if h==2.0:
##              plt.figure(2)
##              # Plot and Figure Formatting
##              plt.plot(nx_e,ny_e,'b',nx_est,ny_est,'g',
##                       nexact_design[0],nexact_design[1],
##                       'ko',linewidth=2.0)
##              ax = plt.gca()
##              ax.set_xlim((xmin,xmax))
##              ax.set_xticks(xspace)
##              ax.set_ylim((ymin,ymax))
##              ax.set_yticks(yspace)
##              ax.get_xaxis().set_visible(False)
##              ax.get_yaxis().set_visible(False)
##              print xspace,yspace
##
##          elif h==1.0:
##              plt.figure(3)
##              # Plot and Figure Formatting
##              plt.plot(nx_e,ny_e,'b',nx_est,ny_est,'g',
##                       nexact_design[0],nexact_design[1],
##                       'ko',linewidth=2.0)
##              ax = plt.gca()
##              ax.set_xlim((xmin,xmax))
##              ax.set_xticks(xspace)
##              ax.set_ylim((ymin,ymax))
##              ax.set_yticks(yspace)
##              ax.get_xaxis().set_visible(False)
##              ax.get_yaxis().set_visible(False)
##              print xspace,yspace
##
          ##########################
          #Variable Update
          ##########################
          Design_norm=dot((X_k-X_d),transpose(X_k-X_d))
          X_k=X_d
          k=k+1

      print coeff

Response_Surface_Gen()
```

Lim_State_Estimation_Inpcorp.py

B.5 FORM Code

```
# Date Last Modified: 10/29/2012

# Function:
# This program performs the FORM simulation. The
# Set_Up_Sim.py program is used to specify the number
# of random variables and associated valus, and is
# shared between mutliple programs. The program uses
# an analytic formulation of the limit state,
# calculates the MPP and through an iterative process
# discussed in the Master's thesis updates the search
# point and MPP based on the limit state evaluations.

# Revisions:

# Program Start
from numpy import *
from numpy import linalg as LA
import sympy as S
from Set_Up_Sim import *

def Lim_State(X_vector,Std_dev,Mean):
    # Returns the evaluation of the limit state
    # function and a matrix of partial
    # derivatives of the limit state function

    del_fun=zeros(shape(X_vector))

    # Limit State definitions specific to cases
    # described in the thesis
    u1,u2=S.symbols('u1,u2')

    #Convert from U to X Space
    x1=u1*Std_dev[0,0]+Mean[0,0]
    x2=u2*Std_dev[0,1]+Mean[0,1]

    # Response Surface Estimates for cases
    #Case 1#
    g_RSM1=(.224488599 +x1*3.08789649
            +x2*-.946320549 +x1**2*-1.95157214
            +x2**2*-3.63622622e-07)

    #Case 2#
##    g_RSM2=(.544723209 +x1*4.90572449e-03
```

```
##                +x2*-4.84190622e-01
##                +x1**2*3.57186312e-04
##                +x2**2*2.32346053e-07)

##     #Case 3#
##     g_RSM3=(-2.78243902e+00 +x1*6.40602208
##                +x2*-.483463606 +x1**2*-3.05118722
##                +x2**2*2.24854583e-06)

    g=g_RSM1 #Toggles
    del_g=[S.diff(g,u1),S.diff(g,u2)]
    fun=g.evalf(subs={u1:X_vector[0,0],u2:X_vector[0,1]})

    del_fun[0,0]=(del_g[0].evalf(subs={u1:X_vector[0,0],
                                       u2:X_vector[0,1]}))
    del_fun[0,1]=(del_g[1].evalf(subs={u1:X_vector[0,0],
                                       u2:X_vector[0,1]}))

    return [fun,del_fun]

def FORM():
    [Mean, Std_dev]= Set_Up()

    U_0=zeros(shape(Mean))
    X_0=U_0; U_new=U_0; del_g_new=U_0;

    res_1=1.0; res_2=1.0; res_3=1.0
    epsilon_1=1e-5; epsilon_2=1e-5; epsilon_3=1e-5

    C1=array([[1.00251263, 1.13591738]]) # Case 1
    C2=array([[1.2691578, 1.05672235]])  # Case 2
    C3=array([[1.14882997, 1.13763627]]) # Case 3
    Update=[C1,C2,C3]

    U_0=Update[2] # Toggle Starting Point
    print U_0

    while ((res_1>epsilon_1)|(res_2>epsilon_2)|(res_3>epsilon_3)):
        # Calculate Limit State and First Partial
        [g_0, del_g_0]=Lim_State(U_0,Std_dev,Mean)

        # Norm of First Partial,
        # Unit Vector of Gradient,
        # and Reliability Index
        mag_del=sqrt(dot(del_g_0,transpose(del_g_0)))
```

```
            a_0=del_g_0/mag_del
            beta_0=dot(U_0,transpose(U_0))**.5

            # Update Values
            U_new=-a_0*(beta_0+g_0/mag_del)
            val_new=Lim_State(U_new,Std_dev,Mean)
            del_g_new=val_new[1]

            # Calculate Residuals
            res_1=(dot(abs(U_new-U_0),transpose(abs(U_new-U_0))))**.5
            res_2=(dot(abs(del_g_0-del_g_new),transpose(abs(del_g_0-
                del_g_new))))**.5
            res_3=abs(g_0/mag_del)
            U_0=U_new

            print 'Update',U_0, 'beta_0',beta_0

        print 'Sensitivity', U_0/beta_0
        return
FORM()
```

MPP_Testing.py

Bibliography

[1] Mark Ardema. *Newton-Euler Dynamics*. Springer, 2005.

[2] C.G. Bucher. A fast and efficient response surface approach for structural reliability problems. *Structural Safety*, 7(1):57–66, January 1990.

[3] Xiaoping Du. *Probabilistic Engineering Design*, chapter 7: First Order and Second Reliability Methods. 2005.

[4] S. Chen F. Chen. Reliability-based assessment of vehicle safety in adverse driving conditions. *Tansportation Research Part C*, 19(1):156–168, February 2011.

[5] J. Snaebjornsson R. Sigbjornsson. Probabilistic assessment of wind related accidents of road vehicles: A reliability approach. *Journal of Wind Engineering and Industrial Aerodynamics*, 74-76(1):1079–1090, April 1998.

[6] M Rajashekhar and B Ellingwood. A new look at the response surface approach for reliability analysis. *Structural Safety*, 12(3):205–220, October 1993.

[7] David Robinson. A Survey of Probailistic Methods Used in Reliability, Risk, and Uncertainty Analysis: Analytical Techniques I. Technical Report SAND98-1189, Sandia National Laboratories, Alburquerque, New Mexico 87185, June 1998.